DATA-DRIVEN UNCERTAINTY ANALYSIS IN NEURAL NETWORKS WITH APPLICATIONS TO MANUFACTURING PROCESS MONITORING

by

Bin Zhang

A Dissertation

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Doctor of Philosophy

School of Mechanical Engineering

West Lafayette, Indiana

August 2021

THE PURDUE UNIVERSITY GRADUATE SCHOOL

STATEMENT OF COMMITTEE APPROVAL

Dr. Yung C. Shin, Chair

School of Mechanical Engineering

Dr. Peter H. Meckl

School of Mechanical Engineering

Dr. Bin Yao

School of Mechanical Engineering

Dr. Stanislaw H. Zak

School of Electrical and Computer Engineering

Approved by:

Dr. Nicole L. Key

ACKNOWLEDGMENTS

First and foremost, I would like to appreciate my major advisor Professor Yung C. Shin for his insightful guidance and enthusiastic support for my Ph.D. program.

Second, I would like to thank my colleagues who provided invaluable help for my studies, including Christopher Katinas for his contribution to the data-driven tool wear monitoring works, Shunyu Liu for her support on the laser additive manufacturing experiments and the preparation of porosity measurement samples, Kyung-Min Hong for his assistance on the laser welding experiments and Phuong Ngo for his help on the neural-fuzzy model training. I would like to also appreciate Prof. Peter H. Meckl, Prof. Bin Yao, and Prof. Stanislaw H. Zak for serving on my advisory committee.

Finally, I'm grateful to my parents and all other family members for their patience and support for my Ph.D. study.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Artificial neural networks, including deep neural networks, play a central role in data-driven science due to their superior learning capacity and adaptability to different tasks and data structures. However, although quantitative uncertainty analysis is essential for training and deploying reliable data-driven models, the uncertainties in neural networks are often overlooked or underestimated in many studies, mainly due to the lack of a high-fidelity and computationally efficient uncertainty quantification approach. In this work, a novel uncertainty analysis scheme is developed. The Gaussian mixture model is used to characterize the probability distributions of uncertainties in arbitrary forms, which yields higher fidelity than the presumed distribution forms, like Gaussian, when the underlying uncertainty is multimodal, and is more compact and efficient than large-scale Monte Carlo sampling. The fidelity of the Gaussian mixture is refined through adaptive scheduling of the width of each Gaussian component based on the active assessment of the factors that could deteriorate the uncertainty representation quality, such as the nonlinearity of activation functions in the neural network.

Following this idea, an adaptive Gaussian mixture scheme of nonlinear uncertainty propagation is proposed to effectively propagate the probability distributions of uncertainties through layers in deep neural networks or through time in recurrent neural networks. An adaptive Gaussian mixture filter (AGMF) is then designed based on this uncertainty propagation scheme. By approximating the dynamics of a highly nonlinear system with a feedforward neural network, the adaptive Gaussian mixture refinement is applied at both the state prediction and Bayesian update steps to closely track the distribution of unmeasurable states. As a result, this new AGMF exhibits state-of-the-art accuracy with a reasonable computational cost on highly nonlinear state estimation problems subject to high magnitudes of uncertainties. Next, a probabilistic neural network with Gaussian-mixture-distributed parameters (GM-PNN) is developed. The adaptive Gaussian mixture scheme is extended to refine intermediate layer states and ensure the fidelity of both linear and nonlinear transformations within the network so that the predictive distribution of output target can be inferred directly without sampling or approximation of integration. The derivatives of the loss function with respect to all the probabilistic parameters in this network are derived explicitly, and therefore, the GM-PNN can be easily trained with any backpropagation method to address practical data-driven problems subject to uncertainties.

The GM-PNN is applied to two data-driven condition monitoring schemes of manufacturing processes. For tool wear monitoring in the turning process, a systematic feature normalization and selection scheme is proposed for the engineering of optimal feature sets extracted from sensor signals. The predictive tool wear models are established using two methods, one is a type-2 fuzzy network for interval-type uncertainty quantification and the other is the GM-PNN for probabilistic uncertainty quantification. For porosity monitoring in laser additive manufacturing processes, convolutional neural network (CNN) is used to directly learn patterns from melt-pool patterns to predict porosity. The classical CNN models without consideration of uncertainty are compared with the CNN models in which GM-PNN is embedded as an uncertainty quantification module. For both monitoring schemes, experimental results show that the GM-PNN not only achieves higher prediction accuracies of process conditions than the classical models but also provides more effective uncertainty quantification to facilitate the process-level decision-making in the manufacturing environment.

Based on the developed uncertainty analysis methods and their proven successes in practical applications, some directions for future studies are suggested. Closed-loop control systems may be synthesized by combining the AGMF with data-driven controller design. The AGMF can also be extended from a state estimator to the parameter estimation problems in data-driven models. In addition, the GM-PNN scheme may be expanded to directly build more complicated models like convolutional or recurrent neural networks.

1. INTRODUCTION

1.1 Background

In the last decade, data science has emerged as a prevalent paradigm in the modeling and analysis of real-world problems in almost every quantitative and scientific discipline [1], such as finance (e.g., fraud detection [2] and stock prediction [3]), healthcare (e.g., medical diagnosis with tomography images [4] and electrocardiography signals [5]) and engineering (e.g., control system design [6], condition monitoring [7] and maintenance [8] of industrial equipment), among others, for which the first-principle-based models are often not available or difficult to establish [9]. Among the numerous data-driven methods studied in the machine learning and deep learning community, artificial neural networks appear to be one of the most popular and versatile classes of models. A neural network uses a layered architecture and interconnected neurons with learnable parameters for information processing. Due to its flexible structure, it can be adapted to different tasks (e.g., regression [10] and classification [11]) and data modalities (e.g., recurrent neural networks for time-series data [12] and convolutional neural networks for image data [13]).

Learning neural network models, as well as other types of machine learning models, always involves addressing uncertainties, including the epistemic uncertainty within the underlying process or system from which data are collected and aleatoric uncertainty introduced during data collection like measurement noise [14]. Therefore, uncertainty analysis is essential for any data-driven modeling effort and the Bayesian machine learning methods that work with probabilistic models have attracted more research interests in recent years [15]. For instance, the probabilistic neural network, as illustrated in Figure 1-1, incorporates the probability distributions of network parameters to quantify uncertainties [16]. It could provide quantitative uncertainty information associated with its predictions by inferring the predictive probability distributions of output and thus exhibits higher reliability than its deterministic counterpart in many fields of applications. The uncertainty analysis in neural networks is valuable for a broad range of data-driven problems. For example, given the neural network's capability of approximating any nonlinear function, it might be used for the data-driven system identification [17], state estimation [18], and control [19] of nonlinear dynamic systems under stochastic uncertainties.

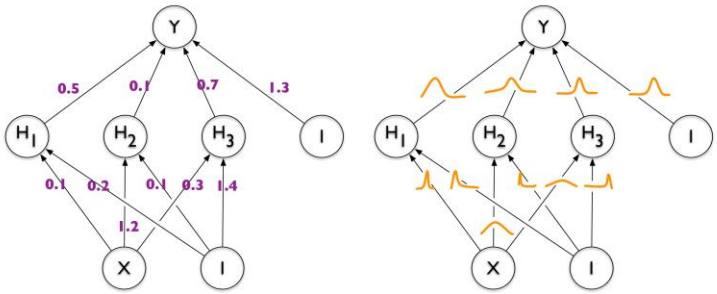


Figure 1-1 An illustration of probabilistic neural networks [16]
(left: a classical neural network with deterministic parameters; right: a probabilistic neural network whose weighting factors have probability distributions)

The health condition monitoring of systems and processes is also a field where data-driven models are extensively used and uncertainty assessment is needed [20][21]. Instead of relying on profound field knowledge to develop first-principle-model-based monitoring schemes [22], the data-driven monitoring methods learn normal and abnormal condition patterns of a system from run-time sensor signal data to develop predictive models for fault detection [23], isolation [24], remaining life prediction [25] and maintenance management [26]. Since the data measurement will be inevitably impacted by noises and the target system may exhibit unpredictable variations during operation, uncertainty quantification is crucial for the data-based monitoring models [27]-[29], with which diagnostic decisions could be made more confidently.

While data-driven condition monitoring has been studied for a variety of systems, this work will focus on the monitoring of manufacturing processes. A manufacturing plant is composed of a variety of machine units handling different material processing and assembling tasks, through which raw materials are transformed to functional products. To keep such a system running with high productivity and consistent product quality, the conditions of the process being operated on each machine unit should be closely monitored such that any abnormality can be detected and eliminated with timely intervention actions before causing any loss. The necessity of condition monitoring might be exemplified with additive manufacturing (AM) processes. In recent decades, laser additive manufacturing, which uses a laser beam to melt metallic powders and then deposits solidified materials layer-by-layer, has been widely used to fabricate high-performance metallic components. However, during the melting-solidification process, a variety of quality defects, such as porosity, cracking and delamination [30][31], may occur in the fabricated parts, which

deteriorates the parts' performance (durability, strength, etc.) and hinders their uses in mission-critical scenarios. Hence, a trial-and-error procedure to determine the optimal defect-free process parameters (e.g., laser power and beam scan speed) is required [32]. Even so, because of the variability of AM processes, defects may still occur unexpectedly, which necessitates expensive and time-consuming post-process measurement for quality inspection and assurance. In a survey in 2014, 47.2% of the 108 manufacturer respondents indicated that the uncertain quality of final products was a barrier to their adoption of AM [33]. Therefore, to ensure product reliability and quality consistency, an in-situ condition monitoring scheme is critically needed for the laser AM processes, as recognized in several recent strategic initiatives [34]-[36].

1.2 Motivation

Condition monitoring is vital for the quality assurance and process-level automation of not only laser AM but also almost all the other manufacturing processes. Therefore, almost all the latest data-driven models have been applied to the predictive modeling of process conditions by different researchers. In a conventional machine learning fashion, features relevant to the process conditions are extracted via signal processing of measurable process data (e.g., power, vibration, acoustic, thermal) and then machine learning models are trained to predict the process conditions based on the selected features. In recent years, deep learning models have also been applied to directly learn patterns from raw measurement data. However, in spite of the numerous efforts that have been made, most manufacturing systems in the industry are still operated in an open-loop manner at the process level. Why the process condition monitoring systems have not been widely deployed can be attributed to multiple reasons, such as the lack of affordable data sensing devices and the insufficiency of predictive models to cover a wide range of operating conditions. Among them, one of the biggest obstacles is that the uncertainties associated with manufacturing processes have not been properly addressed. While the monitoring systems are aimed to make the processes more reliable, the fidelity of a monitoring scheme itself can be questionable, given the variability of manufacturing processes and noise in the industrial environment. For example, a change of material supplier and the batch of tools may cause noticeable variations of the signals from drilling processes under the same operating conditions. As a result, the performance of a tool-wear monitoring model trained for one drilling operation may degrade on its nominally identical counterpart, as described in a case study in [37].

The insufficient consideration of uncertainty is a key bottleneck for not only manufacturing process monitoring but also many other practical applications where neural networks are widely used, such as the design of data-driven state estimators and controllers [38][39][40]. The existing neural network applications with uncertainty-modeling effort are mainly based on interval-type uncertainty quantification [42], while only estimating prediction intervals can not provide useful probability distribution information within the lower and upper bounds. On the other hand, the probabilistic neural network, though has emerged as a popular topic within the machine learning community, has not been widely used in practice. This is mainly because training a probabilistic neural network is much more difficult than training a classical one. Especially, estimating the probability distribution of network parameters in an exact Bayesian scheme is intractable in most cases. Hence, some approximation methods must be utilized and a trade-off between uncertainty quantification fidelity and computational efficiency must be made [43]. In some works, to make the Bayesian inference tractable and efficient, the distributions of network parameters and states are assumed to be in certain forms, such as diagonal Gaussian distribution [16]. However, the true distributions of uncertainty in data can be intricate and multimodal, and those presumed forms may introduce a risk of oversimplifying or underestimating the uncertainty. In contrast, the approximate inference methods based on Monte Carlo sampling, though possess higher fidelity, may be computationally intensive and less scalable to complex models and large datasets [44]. Therefore, a new uncertainty analysis scheme that can characterize general-form probabilistic uncertainties in neural networks with both high fidelity and reasonable computational cost is highly desired. Moreover, if the probabilistic network could be then trained efficiently using the well-developed backpropagation methods without invoking large-scale Monte Carlo sampling, this scheme will be invaluable for a broad range of data-driven problems subject to uncertainties, including the predictive modeling of process conditions for manufacturing process monitoring.

The objective of this study is to develop such an innovative uncertainty analysis scheme for neural networks and explore its application to practical problems. To achieve this, the nonlinearity effects in propagating probability distributions through neural networks should be addressed first, as it is fundamental to the following works. With this uncertainty propagation solution, a neuro-Bayesian filter for state estimation of highly nonlinear systems can be derived and a probabilistic neural network training scheme can be designed. The new probabilistic network will be bench-marked against the state-of-the-art methods and applied to the manufacturing process monitoring

applications. A brief roadmap of this study is provided in Figure 1-2. To more clearly recognize the limits in the current state of the art and set the direction of this study, the existing works in the literature relevant to each of the deliverables in this roadmap are reviewed below.

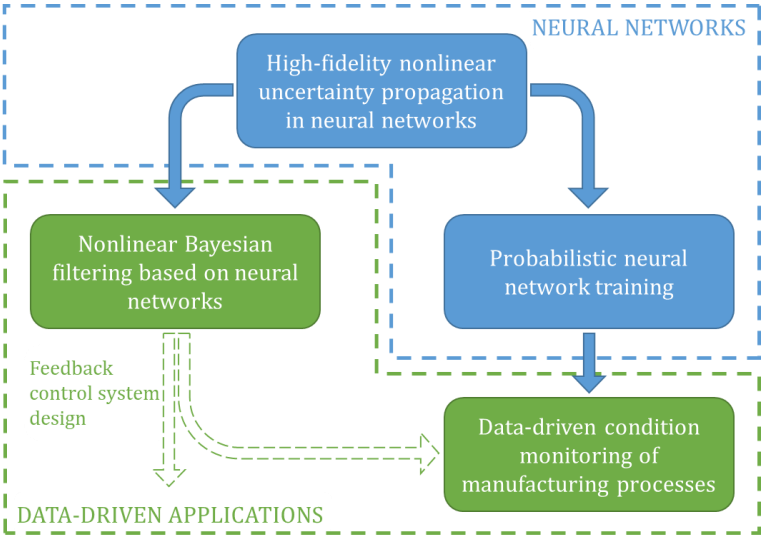


Figure 1-2 A roadmap of this study
(Dashed arrows denote potential works for the future)

1.3 Literature Review

1.3.1 Uncertainty Propagation in Neural Networks

A fundamental challenge in training and deploying a neural network model with uncertainty is the uncertainty propagation through the network, especially through the nonlinear activation functions. Although the propagation of Gaussian uncertainty through linear layers still results in Gaussian distributions, the nonlinear transformation of a Gaussian variable will produce a non-Gaussian distribution that needs to be approximated. This challenge is not only raised by the probabilistic network parameters, but also by the input uncertainty that needs to be propagated forward towards output and the uncertainties in initial conditions and feedback states in a context layer of a recurrent neural network that needs to be propagated to subsequent time steps. Hence, improving the fidelity and efficiency of uncertainty propagation is essential for the inference of predictive distributions in probabilistic neural networks and a variety of uncertainty propagation methods have been proposed in the literature.

For Gaussian uncertainties, the most commonly used propagation paradigm is to estimate the

mean and covariance after nonlinear functions to reconstruct an output Gaussian distribution. This Gaussian-to-Gaussian mapping preserves the form of uncertainty representation and hence can be used for the recursive propagation through time in recurrent networks and layer-wise propagation in deep networks. The linearization method, including local linearization via the first-order Taylor expansion and stochastic linearization by minimizing the expectation of approximation error [45], might be the most intuitive method, though it may not yield accurate results when the magnitude of uncertainty is large. To attain higher fidelity, moment matching solutions have been developed for the specific activation functions in neural networks. Gast et al. [46] developed a variational inference scheme for deep neural networks using the closed-form moment solutions of rectified linear unit (ReLU) and leaky ReLU functions derived by Frey in [47]. Wang et al. [48] derived the analytical solutions of Gaussian distribution moment matching for commonly used activation functions and also discussed the solutions of other exponential-family distributions. Abdelaziz et al. [49] approximated the logistic function with two piecewise exponential functions for uncertainty propagation in deep neural networks. Bui et al. [50] used an approximate expectation propagation scheme to estimate the means and variances in Gaussian processes. Lee et al. [51] derived the covariance kernels in Gaussian processes for the ReLU and hyperbolic tangent functions and implemented the propagation using lookup tables.

Different from the above function-approximation-based methods, there are also sampling-based methods that propagate a set of samples through the nonlinear function as a black box and then evaluate uncertainty from output samples. The unscented transform (UT) in the unscented Kalman filter, which could approximate the output mean and covariance up to the second order accuracy using weighted sigma points [52], is one of the most significant approaches in this category and is often used as the baseline to evaluate other methods. Abdelaziz et al. [49] applied the UT to deep neural networks and achieved better uncertainty estimation accuracy than the exponential function approximation method. The Monte Carlo (MC) simulation that utilizes the random samples drawn from an input distribution to predict the statistic information of output is the most versatile approach, as it is applicable to almost any functions and distributions. Dong et al. [53] applied the MC-based particle filters to the state estimation of battery systems modeled with neural networks. Gal [15] used small MC sets (10~20 samples) to estimate the mean and variance after nonlinear layers in the variational inference through neural networks. However, if it is of interest to predict the output probability distribution instead of just estimating the moments,

then a massive amount of samples needs to be generated and it can be computationally intensive to propagate these samples and reconstruct a probability distribution from the output.

On the contrary, the Gaussian mixture model, which has the capacity to approximate any non-Gaussian distributions [54], could incorporate more information than a single Gaussian distribution and can be used as a more compact uncertainty representation than Monte Carlo samples [55]. To use the Gaussian mixture model in non-Gaussian uncertainty propagation, a high-fidelity output mixture needs to be estimated from the input mixture. Though each Gaussian component can be easily propagated via any moment matching methods, determining the optimal number of components and weights is often a challenge. Terejanu et al. [57] showed that the weight of a Gaussian component can be assumed invariant through a nonlinear mapping only if its variance is infinitesimal. They then proposed two quadratic-programming methods for weight updating through nonlinear systems and applied them to the Gaussian sum filter design [58]. Faubel et al. [59] built an unscented Gaussian mixture filter that recursively splits each Gaussian filter into a number of mixtures to handle non-Gaussian noises and then merges the mixture filter based on estimated posterior density to regulate the number of mixtures. Horwood et al. [60] also developed a similar Gaussian splitting-merging scheme using the Gauss-Hermite quadrature. DeMars et al. [61] used the difference of rates of change of a Gaussian distribution's entropy evaluated by linear and UT methods to measure the level of nonlinearity and split the distribution if the difference was larger than a threshold. Tuggle et al. [62] developed a combined criterion to split a Gaussian distribution along the eigendirection at which the directional derivative of Jacobian (a measure of nonlinearity level) and directional variance (a measure of uncertainty magnitude) are both large.

It can be learned from the above review that 1) the moment matching methods for Gaussian-to-Gaussian propagation could not predict the non-Gaussianity caused by nonlinearity; 2) the non-Gaussian propagation methods using Monte Carlo and Gaussian mixtures, in spite of their higher fidelity of uncertainty representation, could be significantly slower than Gaussian propagation due to the sampling, optimization or recursive splitting routines adopted [57][59], which makes them not practical for deployment in neural networks; 3) the adaptive Gaussian splitting schemes based on the detection of nonlinearity, if combined with a suitable mixture reduction routine, could achieve a high fidelity with a reasonable computational cost; however, none of such schemes has ever been designed for neural networks. In particular, the existing splitting criteria designed for

general nonlinear functions may not yield the best performance for the activation functions in neural networks. Therefore, a new adaptive Gaussian mixture scheme for high-fidelity and efficient uncertainty propagation in neural networks needs to be developed.

1.3.2 Nonlinear Bayesian State Estimation

The estimation of unmeasurable state quantities in the dynamic system subject to stochastic uncertainty is of great importance in a broad range of applications, such as signal processing, positioning and navigation, and control system design [63]. Given a system model, the recursive inference of state information over time from noisy measurements based on Bayesian statistics is referred to as Bayesian filtering [64]. Since Bayesian filtering naturally involves the propagation of state uncertainty, it is a field where the uncertainty propagation scheme mentioned at the end of section 1.3.1 might be applicable, given that such a scheme can be developed. Therefore, the existing Bayesian filtering methods are reviewed below.

For linear systems with Gaussian white noise, the renowned Kalman filter is proved to be the optimal Bayesian state estimator [65]. For nonlinear systems, the extended Kalman filter (EKF) that linearizes the system model and then applies the Kalman filter is a commonly used solution. However, the EKF may produce inaccurate or even divergent results for highly nonlinear systems as it only retains the first-order Tayler expansion [66]. Therefore, a variety of nonlinear versions of the Kalman filter have been developed over the past few decades. The unscented Kalman filter (UKF) utilizes the unscented transform to estimate the moments of nonlinearly transformed state variables at a second-order accuracy [66]. Similarly, the quadrature Kalman filter (QKF) employs the Gauss–Hermite quadrature [67], and the cubature Kalman filter (CKF) uses the cubature rules [68] to evaluate the moment integrals at a higher-order accuracy. Moreover, the ensemble Kalman filter (EnKF) draws random samples from a Gaussian state distribution and applies a Kalman filter to the ensemble of samples [69]. What these nonlinear filters have in common is that the distribution of state variables is assumed to be always Gaussian, while they utilize different integral evaluation techniques to estimate the mean and covariance. However, due to the nonlinearity in the system, the state distribution may exhibit non-Gaussianity like skewness and multimodality, even if the initial state uncertainty is Gaussian. Hence, only assessing the first two moments can not fully characterize the probability density function (PDF) of states and the state estimation result may be suboptimal [70].

For nonlinear and non-Gaussian state estimation problems, the particle filter (PF) is one of the most popular methods [71]. Through sequential importance sampling, the PF can draw random particles from non-Gaussian distributions to represent the state PDF [8], and then the particles are propagated through the system dynamics and reweighted according to measurement likelihood to track the PDF's evolution over time. However, though PF is applicable to almost any system and PDF [70], the number of particles needed to achieve a satisfactory result for a high-dimensional problem could be overwhelming, and consequently, the computational cost might be unaffordable [55][73].

Another non-Gaussian Bayesian estimation paradigm alternative to PF is the Gaussian sum filter (GSF) or Gaussian mixture filter (GMF). Compared with the particles in PF that act as Dirac delta functions with infinitesimal width, a Gaussian mixture component with non-zero covariance contains probability information in a nonnegligible sub-domain of the state space. Therefore, to approximate a PDF to the same degree of accuracy, a Gaussian mixture should need many fewer components than the particles needed by a PF, and thus may resolve the curse of dimensionality [55]. Then for GMF, it is of central importance to determine the Gaussian mixture parameters and ensure the PDF tracking quality. As proved in [73], a GMF approaches the optimal estimation if the covariance of each component is sufficiently small. Nevertheless, using too many narrow components to approximate a PDF will dramatically increase the computational cost and thus a compromise must be made. By reviewing the literature, the existing GMF schemes can be roughly classified into three groups:

1) A fixed number of parallel Gaussian filters. In the conventional GMFs, with the state PDF represented as a Gaussian mixture, a bank of Gaussian filters (e.g., EKF [74] and QKF [75]) can be applied parallelly to each component without altering mixture size. In [74], Sorenson et al. used an Lk norm objective function to optimize the Gaussian mixture approximation. However, even if it is optimized to be amply fine for the initial uncertainty, the components are not guaranteed to retain narrow as they evolve over time, while numerically re-optimizing the entire Gaussian mixture in each time step can be too costly. Hence, Tam et al. [76] proposed a method to analytically update the Gaussian mixture likelihood based on in-process measurement, and Terejanu et al. [58] used a quadratic programming scheme to refine the approximation by only re-optimizing the component weights.

2) Re-approximation via resampling. One way to regulate the widths of covariance in GMF

is to resample a collection of narrow components from the calculated Gaussian mixture to re-approximate the state PDF. Kotecha et al. [77] built a series of particle GMFs and a resampling step was used to prevent the number of PFs from increasing exponentially when combined with Gaussian sum noises. Raihan et al. [78] improved the particle GMF by combining all the particles from different PFs to refit a new Gaussian mixture from the entire particle pool, instead of running PFs parallelly. Stordal et al. [79] derived a GMF from EnKF, in which the ensemble covariance is scaled down by a bandwidth parameter before being assigned to samples so that the attained Gaussian mixture is fine enough to approximate non-Gaussian PDFs. Psiaki [80] created a resampling scheme to generate Gaussian mixture components with bounded covariance matrices [10], based on which a GMF called blob filter was developed.

3) Adaptive splitting of components. The resampling methods will put a universal bound on the covariance so that all the components are uniformly fine. However, since the complexity of state PDF changes over time and varies across the state space, it should be more efficient to refine the Gaussian mixture adaptively, i.e., more and finer components are used when the PDF is complex, while fewer and coarser components can be used when the PDF is simple. The adaptive splitting method is designed to actively assess the impact of nonlinearity such that any distorted component that exhibits nonnegligible non-Gaussianity will be detected and split into narrower sub-components. Although many GMF works in this category have been mentioned in sub-section 1.3.1 as the idea of adaptive Gaussian mixture refinement coincides, these works are briefed below again from a filter design perspective. Faubel et al. [59] built an unscented GMF that recursively splits UKF components until the sigma points are linearly spaced after the measurement function. Horwood et al. [60] proposed to compare the Gaussian moments from the current QKF and a finer one after splitting to decide whether the splitting is needed. Leong et al. [81] presented a cubature GMF that assesses the nonlinearity in measurement function by the deviation of cubature points from a linear fitting. DeMars et al. [61] established an uncertainty propagation scheme that assesses nonlinearity by the difference of Gaussian entropies estimated using a linear and a nonlinear method. Tuggle et al. [62] created a measurement update scheme that splits Gaussian components based on the Kullback–Leibler divergence between the PDFs from an EKF and a second-order EKF.

It can be seen from the above review that a Gaussian mixture refinement scheme is essential to sustain the GMF fidelity, whereas extra computational resources will then be required. Also,

the refinement schemes involve additional hyperparameters, such as the number of components and the upper bound of covariance, whose optimal values are problem-dependent and have to be determined via trial-and-error. For GMFs, how well they can adapt to the varying complexity of state PDF with a practical computational cost and without tedious tunning will be the key metric of their performance. In light of this, the adaptive splitting method should be the most promising solution as it could adapt to the impact of nonlinearity and refine Gaussian mixtures only when necessary. Nevertheless, though state estimation is a two-step procedure in nature, most of the existing splitting schemes are solely designed for the measurement update step [59][60][81][62] with only a few for the state prediction step [61]. Also, many of the quantities used to measure nonlinearity, such as the deviation of sigma points [59] and the discrepancies between two Gaussian PDFs [60][62], are not explicitly related to the non-Gaussianity of distorted PDFs. Although the non-Gaussianity and its induced state estimation error will eventually vanish as these quantities decay to zero, how fast this convergence will be during the splitting process still lacks a quantitative analysis, and the error magnitude of state PDF estimation is not quantified in most studies. If the adaptive Gaussian mixture scheme for uncertainty propagation, as mentioned at the end of section 1.3.1, is developed, it should be helpful in resolving the above issues. Thereby, a new adaptive Gaussian mixture filter could be designed, which will be particularly suitable for addressing highly nonlinear state estimation problems subject to high magnitudes of non-Gaussian uncertainties.

1.3.3 Probabilistic Neural Network for Uncertainty Prediction

In recent years, owing to the fast advancing of computing platforms and training algorithms, neural networks, including deep neural networks, have achieved a remarkable leap of prediction accuracy and extensive successes in the data-driven modeling of real-world problems. However, the uncertainties in the neural network models are often overlooked or underestimated in many studies. Many of the existing neural networks with uncertainty quantification are designed based on interval-type uncertainties [42][82]. Prediction intervals of output targets can be constructed using a variety of methods, such as type-2 fuzzy sets [83], interval-type network parameters [84], and bootstrap of networks [85][86]. However, the prediction intervals may be incompetent in providing essential probability density information, especially if the distribution of output target is skewed or multimodal. In this sense, the probabilistic neural networks (PNN) that incorporate

probability distributions of network parameters to infer predictive distributions are more powerful in providing informative uncertainty quantification [16].

The Bayesian neural network (BNN), as a major class of PNN, provides a methodical way to address probabilistic uncertainties in neural networks. Because the exact Bayesian inference of posterior distributions over neural network parameters is not tractable, the BNNs typically use Markov Chain Monte Carlo (MCMC), variational inference (VI), or expectation propagation (EP) methods to estimate the posterior distributions, given a presumed prior distribution and training dataset [87]. The MCMC methods like Hamilton Monte Carlo [88] have been applied to Bayesian learning in neural networks since the 1990s. Lately, Deng et al. [89] proposed an adaptive MCMC scheme of stochastic gradients to learn sparse deep learning models. Nevertheless, the MCMC methods based on sampling from engineered distributions may have poor scalability to large datasets [90]. Alternatively, the VI methods create an approximation of the true posterior by maximizing the evidence lower bound (ELBO). Graves et al. [91] developed a VI scheme scalable to large datasets, in which diagonal Gaussian posterior is adopted and Monte Carlo integration is used to estimate the intractable loss functions. Blundell et al. [16] created a similar scheme based on Monte Carlo gradients called Bayes by backpropagation. Gal et al. [92] revealed that a neural network with Bernoulli weight dropout is equivalent to a BNN with approximate VI, and a concrete dropout strategy is proposed in [93]. The EP is also an approximate inference method and it's different from VI as it minimizes the Kullback-Leibler divergence between the true and approximate posteriors via moment matching. Lobato et al. [90] designed an EP backpropagation scheme by combining the forward propagation of probabilities with backward propagation of gradients, which is considerably faster than Graves' VI method [91] because there is no Monte Carlo approximation of gradients needed. They then developed an alpha divergence scheme that can be scaled between EP and VI to take advantage of both [94]. Recently, Zhao et al. [95] presented a generalized EP scheme that can learn multimodal posterior distributions of parameters as Gaussian mixtures.

In spite of the recent development, there're still several limitations on the existing BNNs: 1) Many of the training schemes require Monte Carlo sampling to evaluate expectation integrals [89][91], which incurs higher computational cost and poor scalability to large-scale datasets. 2) To make the inference with VI or EP tractable, the posterior distribution of parameters is mostly assumed to be in certain simple forms, such as diagonal Gaussian distribution [91], whereas the

true posterior can be complex and multimodal [95]. 3) The performance of Bayesian learning largely depends on the choice of prior and likelihood functions, which must be selected carefully before or during training [96], and some extra hyperparameters, such as the model precision parameter in Gaussian likelihood function, need to be tuned for each problem. 4) The predictive distribution in VI or EP requires the integration of model predictions over the learned posterior parameter space, typically through the ensemble of model prediction samples drawn from the posterior, which can be computationally inefficient [48].

To address those problems, probabilistic networks that are not in fully Bayesian form have been studied. Wang et al. [48] proposed the natural parameter network (NPN), in which all the weights and neuron states are treated as exponential-family distributions. The forward inference of probabilities via moment matching and backpropagation of gradients for natural parameters were derived analytically such that the NPN model can be trained using any gradient-based method. To achieve faster inference, Gast et al. [46] proposed a lightweight probabilistic network that only approximates the distribution of activation states in hidden neurons but not the weights. Tran et al. [97] proposed a Bayesian layer as a new constitutive layer of deep neural networks so that the uncertainty can be dealt with only in this layer instead of the whole network. Training deep neural networks as Gaussian processes has also been studied, whereas the deep Gaussian processes can only predict the output as multivariate Gaussian distributions [50][51]. To model general-form uncertainty, Pawlowski et al. [98] proposed an implicit weight uncertainty scheme that uses an extra Hypernetwork to generate non-Gaussian distributions for the weights in the main network, which outperformed the Bayes by backpropagation method in [16]. The mixture density network [99] has also been employed to fit the output distributions of deep networks [100] and recurrent networks [101] as Gaussian mixtures, whereas the uncertainties in input and hidden neurons cannot be addressed in these mixture density networks.

Although the studies above all provide invaluable inspirations to improve the probabilistic modeling in neural networks, each of them can mainly resolve only one aspect of the BNN's insufficiencies (e.g., the computational efficiency or the modeling of general-form uncertainties). Moreover, in recent years, the neural networks for deep learning are becoming more complex. For example, Table 1-1 summarizes the state-of-the-art convolutional neural networks (CNN) [102]-[107] in the ImageNet Large Scale Visual Recognition Challenge, a data competition to classify millions of images into 1000 categories [108]. As can be seen, while the classification accuracy

has been improved significantly over the years, the CNN architectures are also growing deeper and deeper, with tens of millions of learnable parameters to train and billions of numeric operations (multiplications and additions) to make one prediction [109]. Transforming a deep neural network with a complex structure into a BNN, or any other type of PNN, by assigning distributions to its parameters to do uncertainty modeling is still challenging with the existing approaches because the high fidelity and high efficiency could not be achieved in a unitive way.

Table 1-1 Summary of benchmark CNNs for the ImageNet challenge

Name	Year	Top-5 error	Layers	Parameters (Millions)	Operations per prediction (Billions) [110]
Alexnet [102]	2012	16.4%	8	61	0.72
Vgg-19 [103]	2014	7.4%	19	144	19.6
GoogleNet [104]	2014	6.7%	22	7	1.58
Resnet-101 [105]	2015	3.6%	101	45	7.6
Squeezenet [106]	2016	14.7%	18	1.23	0.39
Inception-Resnet [107]	2016	3.1%	164	56	13.2

Therefore, a new probabilistic neural network that can be trained and implemented in a manner similar to the classical networks without invoking any computationally intensive method for approximate uncertainty inference should be developed. Especially, if the adaptive Gaussian mixture scheme mentioned at the end of section 1.3.1 is realized, the new network will be able to accurately and efficiently quantify the arbitrary probability distributions of its inputs, parameters, intermediate-layer states, and output predictions, which is a capacity that has never be owned by the existing PNN schemes.

1.3.4 Monitoring of Manufacturing Processes using Data-driven Approaches

The condition monitoring of manufacturing processes is considered as the main application field in this study to implement the probabilistic neural networks because uncertainty analysis is critical for the monitoring systems. Since there are so many manufacturing processes that an exhaustive review of their condition monitoring schemes is impossible, only two representative applications, i.e., tool condition monitoring in machining processes and quality monitoring of laser-based additive manufacturing processes will be reviewed here. These two applications bear common characteristics with other monitoring tasks, and the conclusions drawn from the review

of these two applications will also hold for other processes and conditions.

1.3.4.1 Tool condition monitoring of machining processes

Tool condition monitoring in machining processes is one of the most widely investigated subjects in manufacturing process monitoring owing to the long history and widespread use of machining operations in the industry. A significant amount of research effort has been made in tool condition monitoring over the past several decades [111][112]. Among the various monitoring schemes investigated, tool wear monitoring has gained great interest [113] as tool wear is unavoidable during machining and might negatively impact product quality [114][115]. Thus, knowing the tool wear condition allows for scheduling timely tool changes prior to adverse situations. While tool wear can be measured directly by digital cameras, the direct measurements can only be conducted intermittently when the cutting tool is not engaged with the workpiece. Therefore, the indirect methods, which continuously sense in-process measurable signals and infer the tool wear status using data-driven artificial intelligence models, are more suitable for real-time monitoring [113].

Measurable quantities during machining operation include cutting force [116]-[118], vibration [119]-[122], acoustic emission [123]-[125], and spindle power [126][127], all of which contain potentially useful information for indirect tool wear monitoring. From the raw measured data, signal features can be extracted in the time-domain [116][118][126][127], frequency-domain [119][120], and/or time-frequency (wavelet) domain [119][128][129]. To predict tool wear, many of these signal features can be combined to achieve a more reliable prediction in a scheme referred to as ‘sensor fusion’ [111][130]. However, not every signal feature correlates to tool wear, and instead, a subset of features that best predicts tool wear needs to be selected. Features can be selected by ranking the features based on some criterion independent of the model to eliminate the low-ranking features [116][118] or by using techniques like principal component analysis (PCA) [129][131]. These model-independent feature selections are computationally efficient and scalable to large candidate feature pools, though the performance of subsequent tool wear models may not be optimized. In contrast, features can also be selected by considering the tool wear model using a wrapper or embedded approach [132]. In a wrapper approach, the model is treated as a black box to search the optimal feature subset that minimizes the model prediction error [133]. In an embedded approach, a scaling factor is added for each candidate feature to indicate its relative

importance and the factors are optimized during model training [134]. For model-dependent methods, the size of the candidate feature pool should be relatively small; otherwise, a large quantity of training data is required and the model training will be slow due to the curse of dimensionality [140]. Hence, to find useful features a feature selection scheme that takes advantage of both model-dependent and model-independent feature selections is highly desirable but not found in the available literature.

Taking selected signal features as inputs, a predictive model can be established and must be capable of estimating tool wear in real-time. To build the predictive model, data-driven machine learning techniques are used for training from experimental data so that an in-depth understanding of system physical behaviors is not a prerequisite [144]. Examples of machine learning techniques which have been successfully applied to tool wear monitoring include artificial neural networks [113][116][135][136], fuzzy systems [124][137], neuro-fuzzy systems [138][139], support vector machines [118][140], hidden Markov models [141][142], Gaussian mixture models [143], random forests [144] and belief network [145]. These models are mostly constructed as predictive feature-wear mappings for the regression of progressive tool wear growth or classification of tool wear stage (e.g., sharp, workable, and dull). However, the robustness under uncertainty was not sufficiently considered in these studies although it is vital for the reliability of the monitoring systems.

Considerable uncertainties have been observed in the machining process and reproducibility of wear data is difficult due to many factors, such as variations of material properties, application of cutting fluid, environmental noises, and tool wear measurement errors. Addressing uncertainty in tool wear monitoring with probabilistic models emerged in recent years. Karandikar et al. (2014) used Bayesian updating with discrete grid and Markov Chain Monte Carlo (MCMC) methods to predict the probability distribution of tool life by estimating the constants in the Taylor tool life equation considering parameter uncertainty [146][147]. Akhavan et al. [148] also presented an MCMC-based probabilistic approach to learn the parameters in a tool wear mechanics model with uncertainty term using Bayes' rule. Tracking the progressive development of tool wear with uncertainty intervals has also been studied. Ren et al. [149] used the type-2 fuzzy system to predict tool wear with uncertainty bounds. By formulating state-space models to describe the tool wear dynamics and using measurable features such as the RMS of vibration signals as observation variables, the tool wear could be estimated using extended Kalman filter [150] and particle filter

[151][152]. Especially, the particle filter also enabled the use of Gaussian mixtures to approximate the non-Gaussian initial uncertainty of a process [153]. Akhavan et al. [154] combined the Bayesian-based parameter estimation with an extended Kalman filter to minimize the amount of data required for building the tool wear model for a difficult-to-machine material. Kong et al. [155] trained the support vector regression model as a probabilistic model to predict tool wear with confidence intervals. Though it has been proved that reporting the tool wear in an interval form gives more reliable results than crisp-value predictions [151], these proposed methods are still not able to consider the variance of cutting tools, workpiece materials, and cutting conditions.

In addition, deep learning techniques have also been introduced to tool wear monitoring in recent two years. Signal "images" were constructed by stacking signal data from multiple sensors [156] or by decomposing signals into the time-frequency domain using wavelet transform [157]. Then convolutional neural networks were trained to predict tool wear directly from signal data without explicit feature extraction [156]-[159]. However, the applications of deep learning on tool wear monitoring are still preliminary studies and the performance achieved is not superior to those by conventional machine learning techniques, probably because based on the existing signal processing techniques, all the information embedded in a signal can be fully revealed and hence the explicit feature extraction will be sufficient for signal-based tool wear monitoring. Other than tool wear, the detections of chatter [160]-[163] and tool breakage [164][165] have also been studied for the monitoring of machining processes. Since these failures will result in distinct signal patterns upon occurrence, rather than the gradual signal changes due to tool wear, their detections should be relatively easier than tool wear monitoring, and thus will not be considered in this work.

It can be concluded from the above review that the tool wear monitoring systems have only achieved limited success in the industry mainly due to: 1) the insufficient generalizability to the cutting tools, workpiece materials and cutting conditions different from those in the training data, 2) the absence of systematic selection of optimal feature sets; 3) the insufficient consideration of uncertainties in the complicated machining environment. All these issues, in particular the last one, will be addressed in this study.

1.3.4.2 Quality monitoring of additive manufacturing processes

Given that additive manufacturing (AM) is widely used in fabricating metallic components, there have been numerous research efforts that address the monitoring of AM processes for quality

assurance [35][36]. Laser-based AM of metallic parts requires high laser power intensity at the laser-material interaction area, which prevents the use of traditional contact sensors. Therefore, non-contact sensors are mostly used to collect high-dimensional process data with spatial-temporal patterns. The melt-pool monitoring appears to be the most prevalent monitoring paradigm because of its suitability for online measurement. Sensing the thermographic and morphological information of melt pool by instrumenting laser AM systems with sensors might date back to the 1990s [166][167] and has been consistently pursued [168]-[170]. The melt-pool radiation intensity, temperature, and geometry (e.g. area, length, or width) could be measured by a photodetector, an infrared (IR) thermal camera, or a high-speed vision camera, and then used to predict a variety of quality attributes, such as layer thickness [171], thermal deformation [172], build condition [173] and particularly porosity.

For porosity monitoring, Krauss et al. [171] found that a bossing pattern in the irradiance signal is indicative of sublayer cavities (>100 μm). Barua et al. [174] used the deviation of melt-pool temperature gradient from a reference defect-free cooling curve to predict defects like crack or porosity. Clijsters et al. [175] mapped the melt-pool area data to 3-D positions on the specimen and then used a thresholding method to locate pores larger than 100 μm. Mireles et al. [176] also used a thresholding method on the temperature obtained from IR images to detect pores (600~900 μm). Zenzinger et al. [177] presented an optical tomography method to construct a 3-D mapping of possible structural defects by recording the hot and cold spots of melt-pool radiation. Toeppel et al. [178] demonstrated a commercial monitoring system called QM Meltpool 3D, which can detect pores larger than 1/10 mm using an IR camera and a photodetector. Another monitoring module called ThermaViz was demonstrated in [179] for melt-pool temperature and size measurement using a pyrometer and an IR sensor. Khanzadeh et al. processed the data from the ThermaViz system using unsupervised learning methods such as multilinear principal component analysis (MPCA) [180] and self-organizing map (SOM) [181] for automatic feature extraction. The former achieved a defect-detection accuracy of 90.97%, while the latter achieved a detection accuracy of 96.07% for pores ranging from 0.05 to 0.93 mm, but the SOM method was validated only on a single data set. As can be seen, the porosity monitoring schemes in the literature are mostly addressing large porosity defects above 100 μm.

Other than melt-pool-based porosity monitoring, Zhang et al. [182] investigated the track width classification using melt-pool data and achieved an accuracy of 90.1% by a support vector

machine (SVM) with artificially defined features and 92.8% by a convolutional neural network (CNN) with automatically learned features. Gao et al. [183] presented a two-camera scheme for droplet transition and keyhole status monitoring using FFT-based image denoising and statistical feature extraction. Xiong et al. [184] used two cameras (top-view and side-view) focused behind the melt pool to measure track width and height. Slotwinski et al. [185] developed an ultrasonic sensor for off-line porosity measurement. Rieder et al. [186] developed an ultrasonic setup for on-line measurement of track height. Zhang et al. [187] presented an in-situ surface topography measurement method using fringe projection. DePond et al. [188] presented an optical coherence tomography method for in-situ measurement of layer surface roughness. In general, the non-melt-pool monitoring schemes are less suitable for in-process metrology due to the intervention of the process laser beam and melt-pool radiations.

Furthermore, the in-process quality monitoring of laser welding has been investigated. The laser welding process is governed by similar melt-pool physics as laser AM processes and hence its monitoring methods will also be valuable for the laser AM. Ancona et al. [189] found that the mean and standard deviation of plasma's electron temperature calculated from spectrometer data has a correlation to weld defects like lack of penetration and weld disruptions. Nicolosi et al. [190] described a camera-based full penetration extraction method using cellular neural network. Kim et al. [191] presented a keyhole area estimation method based on coaxial image sensing. You et al. found that with the fusion of photodiode and camera data, it was possible to classify sound and defective weldings [192]. By using the wavelet packet decomposition and principal component analysis for feature extraction, neural networks and support vector machines were then built to detect defects like blowouts and undercut, which achieved an accuracy of 81.34% [193]. Luo et al. [194] developed a coaxial camera setup for weld-pool boundary extraction and keyhole size measurement, and with the use of a radial basis function network, it was possible to predict the keyhole penetration depth and inclination angle. A correlation between porosity and keyhole size was also observed, which revealed that the coaxial images could be potentially used for porosity prediction [195]. Xu et al. [196] have also found that the large fluctuation of keyhole was responsible for the pore formation. However, an in-process porosity monitoring scheme for laser welding hasn't been developed in the literature.

Based on in-process data sensing, the closed-loop control of laser AM processes has also become an area of interest. Via building an empirical [197][198] or physics-based [199] model

between the melt-pool temperature and controllable process parameters, mainly the laser power, PID [189], internal-model-principle [197], model predictive control [198] and feedback linearization [199] controllers can be designed to maintain the melt-pool temperature at a constant level. Closed-loop control has also been used to regulate the melt-pool height [199] and width [200]. Besides, the microstructures in deposited materials can also be controlled indirectly, via regulation of melt-pool size [201] and cooling rate [202].

It can be learned from the above review that: 1) even in the latest monitoring works, the quality attributes that can be surveilled are still the deposition geometry like height and width, indirect attributes like melt-pool temperature and size, and large interior defects like cavities and porosities over 100 μm, but the detection of pores smaller than 100 μm is still not available; 2) the features used for monitoring are still mostly conventional melt-pool features like area, aspect ratio and temperature gradient; the automatic feature learning techniques, such as SOM and CNN, have only been adopted in recent years; 3) Because of the lack of reliable monitoring schemes to detect interior quality defects, the process control schemes are mostly indirect schemes designed to maintain a constant melt-pool size or temperature, instead of directly correct defects and assure satisfactory product quality.

Therefore, a process monitoring scheme that could reliably detect interior defects like micro porosity needs to be developed, as such a scheme may enable direct process control for quality assurance. The deep learning models that can automatically learn suitable melt-pool patterns from data are the most promising solution to achieve this goal. However, the porosity may only produce a limited impact on the melt pool, and the accurate post-process measurement of porosity is difficult. Consequently, both the input (melt-pool images) and output (porosity measured from fabricated specimens) data to train the deep learning model may be corrupted by noises, and uncertainty analysis is essential for the monitoring scheme, which can be potentially solved by the probabilistic neural network discussed in section 1.3.3.

1.4 Research Objectives

This study aims to develop an innovative uncertainty analysis scheme for neural networks and explore its application on practical problems like manufacturing process monitoring. To achieve this goal, the study will be carried out with the following breakdown tasks:

I. An adaptive Gaussian mixture scheme for nonlinear uncertainty propagation in neural

networks will be developed first as the foundation of all the following works. To achieve high-fidelity uncertainty propagation, a new nonlinearity assessment criterion customized for the commonly used activation functions and a set of high-precision splitting schemes to refine Gaussian distributions will be established.

II. An adaptive Gaussian mixture filter for Bayesian state estimation will be designed. By modeling the nonlinear system dynamics with a neural network, the adaptive Gaussian mixture scheme will be extended to refine the estimated state distributions at both the prediction and measurement update stages of Bayesian filtering. A quantitative analysis of the state estimation error will also be provided.

III. A probabilistic neural network with Gaussian mixture distributions of parameters will be developed. Based on adaptive layer-wise refinement of Gaussian mixtures, the analytical inference of predictive distributions will be derived and the gradients for backpropagation training will be formulated. This probabilistic network will be benchmarked against the state-of-the-art methods in the literature.

IV. Application of the proposed probabilistic neural network will be made on the condition monitoring of manufacturing processes.

    a. A tool wear monitoring system for the turning process will be developed based on features extracted from sensor signals. The predictive tool wear model built with the new probabilistic network will be compared with the model trained using a classical network with interval-type uncertainty.

    b. A porosity monitoring system for the laser additive manufacturing process will be developed using deep learning to directly learn features from raw camera image data. The porosity prediction performances of convolutional neural networks with and without incorporating the probabilistic network module will be compared.

In brief, the core of this study will be the adaptive Gaussian mixture scheme and the final outcome will be a unitive set of data-driven uncertainty analysis tools for uncertainty propagation, Bayesian filtering and probabilistic network modeling, with demonstrated successes on manufacturing applications.

1.5 Dissertation Outline

The research objectives listed above have been pursued progressively. This dissertation

demonstrates the accomplished research with the following six chapters:

Chapter 1 Introduction:

This chapter briefs the background and motivation of this study. The necessity of data-driven uncertainty analysis for neural networks and its importance on the condition monitoring of manufacturing processes are highlighted. A brief literature review of relevant topics, including nonlinear uncertainty propagation, nonlinear Bayesian filtering, probabilistic neural network modeling and manufacturing process monitoring, is provided to identify the challenges that have not been sufficiently addressed and position our research proposals with respect to the existing works. A breakdown of research objectives is also outlined.

Chapter 2 Adaptive nonlinear uncertainty propagation in neural networks:

In this chapter, a Gaussian-mixture-based uncertainty propagation scheme is developed. The propagation of uncertainty through a network's nonlinear layers is usually a bottleneck because the existing techniques designed to transmit Gaussian distributions via moment estimation are not capable of predicting non-Gaussian distributions. Given that any input uncertainty can be characterized as a Gaussian mixture with a finite number of components, the developed scheme actively examines each mixture component and adaptively splits those whose representation fidelity of uncertainty might be deteriorated by the network's nonlinear activation layer. A novel Kullback–Leibler nonlinearity detection criterion is derived to trigger splitting. Five nonlinear uncertainty propagation examples are presented, in all of which the developed scheme exhibits a high fidelity and efficiency in predicting the evolution of non-Gaussian distributions through recurrent and multi-layer neural networks.

Chapter 3 Adaptive Gaussian mixture filter for nonlinear state estimation:

In this chapter, an adaptive Gaussian mixture filter is developed. The state estimation of highly nonlinear dynamic systems is difficult because the probability distribution of their states can be highly non-Gaussian. To address this problem, the adaptive Gaussian mixture scheme in Chapter 2 is extended to the filter proposed in this chapter, so that the Gaussian mixture models can be refined based on the system's local severity of nonlinearity to achieve a high-fidelity estimation of the state distribution. Nonlinearity assessment criteria are designed to trigger the splitting of Gaussian components at both the prediction and update stages of Bayesian filtering

and the error bound of estimated state distribution is established. The new filter is benchmarked against the existing methods on multiple challenging problems and it consistently offers among-the-best accuracy with a rational computational cost, which proves that it can be used as a reliable state estimator for highly nonlinear systems subject to high magnitudes of uncertainties.

Chapter 4 Probabilistic neural network for uncertainty prediction:

In this chapter, a probabilistic neural network with Gaussian-mixture distributed parameters is developed. Modeling the uncertainty from data is an essential quest in the learning of neural network models but has not been well addressed. The probabilistic network proposed in this chapter provides an efficient and high-fidelity solution for learning multimodal uncertainties in neural networks. The adaptive Gaussian mixture scheme from Chapter 2 is adopted to refine the Gaussian mixture probability distributions and ensure the fidelity of uncertainty propagation in both linear and nonlinear transformations through the network. As its predictive distribution can be inferred analytically, this probabilistic network can be trained efficiently via backpropagation based on any gradient descent algorithm. The proposed network achieves a state-of-the-art performance when benchmarked on a series of public datasets.

Chapter 5 Application on condition monitoring of manufacturing processes:

This chapter describes the applications of the probabilistic neural network in Chapter 4 on two manufacturing process monitoring schemes. First, a tool wear monitoring scheme for turning processes is developed. A systematic feature normalization and selection procedure is proposed to eliminate the features' dependence on cutting conditions, cutting tools and work materials and select the features that have the best performance in predicting tool wear. Two tool wear models are trained, one using a type-2 fuzzy network for interval uncertainty quantification and the other using the network proposed in Chapter 4 for probabilistic uncertainty quantification. Secondly, a porosity monitoring scheme for laser AM processes is developed. A High-speed digital camera is mounted coaxially to the process laser beam for in-process melt-pool sensing and convolutional neural network models are designed to directly learn melt-pool features to predict the porosity attributes. The classical CNN models without uncertainty quantification are compared with the CNN models in which the probabilistic network is incorporated as an uncertainty quantification module. Experimental results show that for both monitoring schemes, the probabilistic network

not only achieves higher prediction accuracies of process conditions than the classical models but also provides more effective uncertainty quantification to facilitate the process-level decision making and intervention in the manufacturing environment.

Chapter 6 Conclusions and Future Work:

This chapter summarizes all the works in the preceding chapters. Some directions for future studies based on the accomplished works are suggested. The adaptive Gaussian mixture filter can be combined with feedback controller design to regulate closed-loop systems. The filter may also be extended from state estimation to parameter estimation problems in data-driven models. In addition, the probabilistic neural network framework, though mainly formulated for multi-layer feedforward neural networks in this study, may be extended to more complicated models like convolutional or recurrent neural networks.

2. ADAPTIVE NONLINEAR UNCERTAINTY PROPAGATION IN NEURAL NETWORKS

This chapter aims to develop an adaptive Gaussian mixture scheme for high-fidelity and computationally efficient uncertainty propagation in neural networks. A criterion for nonlinearity detection is designed by using the properties of activation functions to simplify the computation so that a Gaussian mixture can be propagated through a nonlinear activation function layer with the adaptive splitting operation triggered for each component only when the level of nonlinearity deteriorates fidelity. The developed scheme is validated on three examples of nonlinear dynamic systems and two data-driven examples, in all of which a high fidelity in forecasting non-Gaussian probability distributions is achieved with satisfactory computational efficiency. This adaptive Gaussian mixture scheme will be fundamental to the subsequent works.

2.1 Problem Formulation and Preliminaries

This chapter is to design an uncertainty propagation scheme so that the probability density function (PDF) of a neural network's states passing through a nonlinear activation layer can be accurately predicted as a compact Gaussian mixture model. Then by using the neural network as a function approximator and the Gaussian mixture as a PDF approximator, the Gaussian mixture mapping can be applied repetitively to address a wide range of uncertainty propagation problems, such as the temporal uncertainty evolution in recurrent neural networks and layer-wise uncertainty propagation in a deep neural network. For simplicity and without loss of generality, the formulation in this chapter starts from the two-layer feedforward neural network, which could approximate any nonlinear functions given a sufficient number of hidden nodes [203]. A feedforward neural network with *m* input, *n* output, and *k* hidden nodes can be expressed as:

$$\mathbf{y} = \mathbf{W}_2\mathbf{f}\left(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1\right) + \mathbf{b}_2 \tag{2.1}$$

where $\mathbf{x} \in \mathbb{R}^m$ is the vector of input variables, $\mathbf{y} \in \mathbb{R}^n$ is the vector of output variables, $\mathbf{W}_1 \in \mathbb{R}^{k\times m}$ and $\mathbf{b}_1 \in \mathbb{R}^k$ are the weight and bias in the hidden layer, $\mathbf{W}_2 \in \mathbb{R}^{n\times k}$ and $\mathbf{b}_2 \in \mathbb{R}^n$ are the weight and bias in the output layer and $\mathbf{f}$ is the activation function. Assume that the input $\mathbf{x}$ is subject to uncertainty whose PDF can be represented by a Gaussian mixture:

$$
\begin{aligned}
&p(\mathbf{x}) = \sum_{i=1}^{M} w_{\mathbf{x}}^{i} p_{\mathbf{x}}^{i}(\mathbf{x}) \quad \text{with} \quad \sum_{i=1}^{M} w_{\mathbf{x}}^{i} = 1 \\
&p_{\mathbf{x}}^{i}(\mathbf{x}) = \mathcal{N}\left(\mathbf{x} \mid \boldsymbol{\mu}_{\mathbf{x}}^{i}, \boldsymbol{\Sigma}_{\mathbf{x}}^{i}\right) = \frac{1}{\sqrt{(2\pi)^{m} \det\left(\boldsymbol{\Sigma}_{\mathbf{x}}^{i}\right)}} \exp\left(-\frac{1}{2}\left(\mathbf{x}-\boldsymbol{\mu}_{\mathbf{x}}^{i}\right)^{T} \boldsymbol{\Sigma}_{\mathbf{x}}^{i\,-1}\left(\mathbf{x}-\boldsymbol{\mu}_{\mathbf{x}}^{i}\right)\right)
\end{aligned}
\tag{2.2}
$$

where M is the number of components, $w_{\mathbf{x}}^{i} \in [0,1]$ and $p^{i}(\mathbf{x})$ are the weight and density of the i-th component, and $\boldsymbol{\mu}_{\mathbf{x}}^{i} \in \mathbb{R}^{m}$ and $\boldsymbol{\Sigma}_{\mathbf{x}}^{i} \in \mathbb{R}^{m \times m}$ are the mean and covariance of the i-th component, respectively. Given a sufficient number of components, a Gaussian mixture can be trained using the expectation-maximization algorithm to approximate any general non-Gaussian PDFs [54].

To propagate uncertainty through the network, a linear transformation will be first applied to map the input PDF $p(\mathbf{x})$ to the PDF of pre-activation states $p(\mathbf{z})$:

$$
\begin{aligned}
&\mathbf{z} = \mathbf{W}_1 \mathbf{x} + \mathbf{b}_1 \\
&p(\mathbf{z}) = \sum_{i=1}^{M} w_{\mathbf{z}}^{i} \mathcal{N}\left(\mathbf{z} \mid \boldsymbol{\mu}_{\mathbf{z}}^{i}, \boldsymbol{\Sigma}_{\mathbf{z}}^{i}\right)
\end{aligned}
\quad \Rightarrow \begin{cases}
w_{\mathbf{z}}^{i} = w_{\mathbf{x}}^{i} \\
\boldsymbol{\mu}_{\mathbf{z}}^{i} = \mathbf{W}_1 \boldsymbol{\mu}_{\mathbf{x}}^{i} + \mathbf{b}_1 \\
\boldsymbol{\Sigma}_{\mathbf{z}}^{i} = \mathbf{W}_1 \boldsymbol{\Sigma}_{\mathbf{x}}^{i} \mathbf{W}_1^{T}
\end{cases}
\tag{2.3}
$$

where $\mathbf{z} \in \mathbb{R}^{k}$ is the vector of the network's states before activation function. Similarly, the PDF of post-activation states $p(\mathbf{s})$ can be mapped to the output PDF as

$$
\begin{aligned}
&\mathbf{s} = \mathbf{f}(\mathbf{z}) \quad p(\mathbf{s}) = \sum_{i=1}^{N} w_{\mathbf{s}}^{i} \mathcal{N}\left(\mathbf{s} \mid \boldsymbol{\mu}_{\mathbf{s}}^{i}, \boldsymbol{\Sigma}_{\mathbf{s}}^{i}\right) \\
&\mathbf{y} = \mathbf{W}_2 \mathbf{s} + \mathbf{b}_2 \\
&p(\mathbf{y}) = \sum_{i=1}^{N} w_{\mathbf{y}}^{i} \mathcal{N}\left(\mathbf{y} \mid \boldsymbol{\mu}_{\mathbf{y}}^{i}, \boldsymbol{\Sigma}_{\mathbf{y}}^{i}\right)
\end{aligned}
\quad \Rightarrow \begin{cases}
w_{\mathbf{y}}^{i} = w_{\mathbf{s}}^{i} \\
\boldsymbol{\mu}_{\mathbf{y}}^{i} = \mathbf{W}_2 \boldsymbol{\mu}_{\mathbf{s}}^{i} + \mathbf{b}_2 \\
\boldsymbol{\Sigma}_{\mathbf{y}}^{i} = \mathbf{W}_2 \boldsymbol{\Sigma}_{\mathbf{s}}^{i} \mathbf{W}_2^{T}
\end{cases}
\tag{2.4}
$$

where $\mathbf{s} \in \mathbb{R}^{k}$ is the vector of states after activation function and N is the number of components in the post-activation mixture. In both mappings, the linear transformation of Gaussian variables is utilized to propagate the mean and covariance of each Gaussian component, while the weight and number of components are invariant. Despite this, the nonlinear PDF mapping from pre-activation states to post-activation states is typically the bottleneck:

$$
p(\mathbf{z}) = \sum_{i=1}^{M} w_{\mathbf{z}}^{i} \mathcal{N}\left(\mathbf{z} \mid \boldsymbol{\mu}_{\mathbf{z}}^{i}, \boldsymbol{\Sigma}_{\mathbf{z}}^{i}\right) \xrightarrow{\mathbf{s}=\mathbf{f}(\mathbf{z})} p(\mathbf{s}) = \sum_{i=1}^{N} w_{\mathbf{s}}^{i} \mathcal{N}\left(\mathbf{s} \mid \boldsymbol{\mu}_{\mathbf{s}}^{i}, \boldsymbol{\Sigma}_{\mathbf{s}}^{i}\right)
$$

In general, a pre-activation Gaussian PDF will become non-Gaussian after transformation by a nonlinear activation function. If the function is monotonic and differentiable, then the non-Gaussian PDF of post-activation states can be expressed as [204]:

$$p(\mathbf{s}) = \left| \frac{d\mathbf{z}}{d\mathbf{s}} \right| p(\mathbf{z}) \tag{2.5}$$

where | | denotes the matrix determinant. For example, a Gaussian variable transformed with the logistic function will have a logit-norm distribution, which may be skewed or multimodal and doesn't have analytical solutions of mean and covariance [205]. This type of transform is only suitable for propagating the uncertainty once because the PDF is not maintained in Gaussian form and thus it is not as tractable as a Gaussian mixture for moment extraction and repetitive propagations in the following layers. Therefore, the goal of our uncertainty propagation scheme is to find a Gaussian mixture parameter set $\left\{ w_{\mathbf{s}}^{i}, \boldsymbol{\mu}_{\mathbf{s}}^{i}, \boldsymbol{\Sigma}_{\mathbf{s}}^{i} \right\}$ to approximate the post-activation PDF in Eq. (2.5). In this work, the approximation quality is assessed by the Kullback–Leibler (KL) divergence, which is a commonly used measure of the difference between two PDFs [206]:

$$D_{KL}\left(p \parallel \hat{p} \right) = \int_{-\infty}^{\infty} p(\mathbf{s}) \log \frac{p(\mathbf{s})}{\hat{p}(\mathbf{s})} d\mathbf{s} \tag{2.6}$$

where $p(\mathbf{s})$ is the true post-activation PDF and $\hat{p}(\mathbf{s})$ is its Gaussian mixture approximation. The KL divergence is always non-negative and converges to zero only if the approximating PDF equals the true one, i.e., $\hat{p}(\mathbf{s}) = p(\mathbf{s})$.

Propagating each Gaussian component using local linearization or unscented transform (UT) while assuming the weights and number of components to be invariant could not attain accurate results except when all the covariances are infinitesimal [57]. Therefore, the idea in this work is to recursively split each Gaussian component in the pre-activation mixture until the covariances of all the components become small enough so that the activation function is nearly linear in their spans and their post-activation PDFs are nearly Gaussian. To facilitate this splitting scheme, the activation functions are assumed to be piecewise differentiable and monotonic, so that the true post-activation PDF can be derived using Eq. (2.5) to assess its proximity to a Gaussian PDF. Moreover, only the single fold activation functions are considered so that its Jacobian matrix is always diagonal and the determinant in Eq. (2.5) can be simplified as $\left| d\mathbf{z}/d\mathbf{s} \right| = \prod_{i=1}^{k} dz_i / ds_i$.

Most of the activation functions in neural networks satisfy all these assumptions, though some functions such as softmax and radial basis function have to be excluded. Four activation functions that are most frequently used in neural networks and their derivatives (for computing the Jacobian matrix) are listed in Table 2-1, which include the logistic function, hyperbolic tangent function

(tanh), rectified linear unit (ReLU) and leaky ReLU. The graphs of the functions and their transformed PDFs from Gaussian distribution input are shown in Figure 2-1. As can been seen, the post-activation PDF can be highly non-Gaussian and hence a high-fidelity Gaussian mixture approximation method needs to be derived, as discussed in the next section.

Table 2-1 Activation functions considered in this work

	Logistic	Tanh	ReLU	Leaky ReLU
$f(x)$	$\frac{1}{1+e^{-x}}$	$\frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$	$\begin{cases} 0 & x \le 0 \\ x & x > 0 \end{cases}$	$\begin{cases} cx & x \le 0,\ 0 < c \ll 1 \\ x & x > 0 \end{cases}$
$f'(x)$	$f(x)\left(1-f(x)\right)$	$1-f(x)^{2}$	$\begin{cases} 0 & x \le 0 \\ 1 & x > 0 \end{cases}$	$\begin{cases} c & x \le 0 \\ 1 & x > 0 \end{cases}$

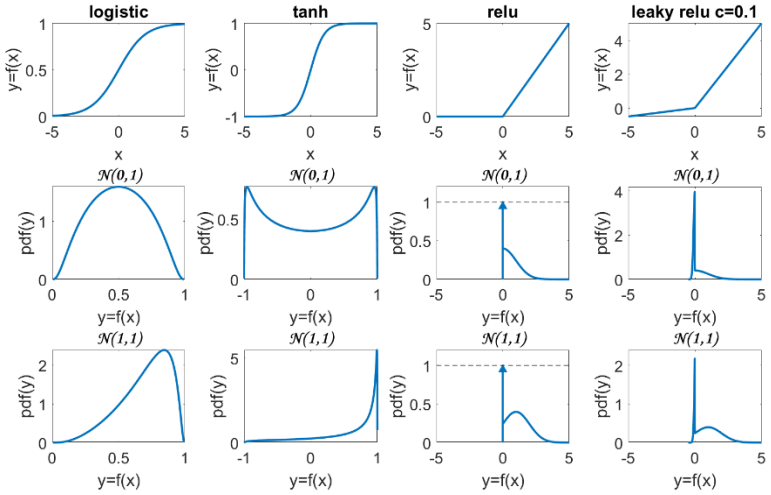


Figure 2-1 Activation functions and their transforms of Gaussian distributions

1st row: graphs of functions, 2nd row: transformed PDFs of a Gaussian input $\mathcal{N}(x|0,1)$, 3rd row: transformed PDFs of a Gaussian input $\mathcal{N}(x|1,1)$

2.2 Methodologies

To establish the adaptive Gaussian mixture scheme for nonlinear uncertainty propagation in neural networks, a nonlinearity detection criterion is proposed first to examine whether a pre-activation Gaussian mixture component will be significantly distorted by the nonlinear activation function. If yes, the detected component will be split into narrower sub-components until the level of nonlinearity will not deteriorate the Gaussian propagation fidelity. Finally, a Gaussian mixture

reduction method is applied to reduce the number of post-activation mixture components. The algorithm details are elaborated in the following subsections.

2.2.1 A Kullback–Leibler Criterion for Nonlinearity Detection

In a neural network, if the pre-activation PDF before a nonlinear activation layer is given as an *M*-component Gaussian mixture, the true post-activation PDF can be derived using Eq. (2.5) as $p(\mathbf{s}) = \sum_{i=1}^{M} w_{\mathbf{s}}^{i} p^{i}(\mathbf{s})$. By applying a certain propagation method to each Gaussian component, an approximating PDF can be obtained as $\hat{p}(\mathbf{s}) = \sum_{i=1}^{M} w_{\mathbf{s}}^{i} \hat{p}^{i}(\mathbf{s})$. Substitute $p(\mathbf{s})$ and $\hat{p}(\mathbf{s})$ into Eq. (2.6), although the KL divergence doesn't have an analytical solution [206], an upper bound can be found using the log sum inequality ($a \log \frac{a}{b} \le \sum_{i=1}^{n} a_i \log \frac{a_i}{b_i}$ with $a = \sum_{i=1}^{n} a_i$ and $b = \sum_{i=1}^{n} b_i$):

$$
\begin{aligned}
D_{KL}\left(p \,\|\, \hat{p}\right) &= \int_{-\infty}^{\infty} \sum_{i=1}^{M} w_{\mathbf{s}}^{i} p^{i}(\mathbf{s}) \log \frac{\sum_{i=1}^{M} w_{\mathbf{s}}^{i} p^{i}(\mathbf{s})}{\sum_{i=1}^{M} w_{\mathbf{s}}^{i} \hat{p}^{i}(\mathbf{s})} d\mathbf{s} \\
&\le \sum_{i=1}^{M} w_{\mathbf{s}}^{i} \int_{-\infty}^{\infty} p^{i}(\mathbf{s}) \log \frac{w_{\mathbf{s}}^{i} p^{i}(\mathbf{s})}{w_{\mathbf{s}}^{i} \hat{p}^{i}(\mathbf{s})} d\mathbf{s} \\
&= \sum_{i=1}^{M} w_{\mathbf{s}}^{i} D_{KL}\left(p^{i} \,\|\, \hat{p}^{i}\right)
\end{aligned}
\tag{2.7}
$$

An upper bound of the overall KL divergence between two mixture PDFs is the weighted sum of the divergence between each component pair. Therefore, a practical strategy to minimize the overall KL divergence is to minimize the divergences of each pair. If the covariance of the *i*-th Gaussian component is so small that the activation function is almost linear within its span, then the true transformed density p^i will be nearly Gaussian and its approximation $\hat{p}^i$ via linearization should be sufficient to achieve a small KL divergence. Hence, the KL divergence between the true transformed PDF and its approximation via linearization is a measure of how much a Gaussian PDF will be distorted by the nonlinearity of activation function and will be used as the criterion to examine whether a Gaussian distribution needs to be split.

To formulate the KL-divergence-based nonlinearity criterion, a single Gaussian distribution is considered as the pre-activation PDF, such that:

$$p(\mathbf{z}) = \mathcal{N}(\mathbf{z} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) \quad \mathbf{z} = [z_1, z_2, \ldots, z_k]^T$$

The true transform $\xrightarrow{\mathbf{s}=\mathbf{f}(\mathbf{z})}$

$$\mathbf{s} = \mathbf{f}(\mathbf{z}) = [f(z_1), f(z_2), \ldots, f(z_k)]^T$$

$$p(\mathbf{s}) = \left(\left| \frac{d\mathbf{z}}{d\mathbf{s}} \right| p(\mathbf{z}) \right)_{\mathbf{z}=\mathbf{f}^{-1}(\mathbf{s})} = \left| \frac{d\mathbf{f}^{-1}(\mathbf{s})}{d\mathbf{s}} \right| \mathcal{N}(\mathbf{f}^{-1}(\mathbf{s}) \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) \tag{2.8}$$

The local linearization $\xrightarrow{\mathbf{s} \approx \mathbf{f}(\boldsymbol{\mu}) + \mathbf{A}(\mathbf{z}-\boldsymbol{\mu})}$

$$\hat{p}(\mathbf{s}) = \mathcal{N}(\mathbf{s} \mid \mathbf{f}(\boldsymbol{\mu}), \mathbf{A}\boldsymbol{\Sigma}\mathbf{A}^T) \quad \mathbf{A} = \left[d\mathbf{f}(\mathbf{z}) \Big/ d\mathbf{z} \right]_{\mathbf{z}=\boldsymbol{\mu}}$$

where $\mathbf{f}^{-1}$ denotes the inverse of function and k is the dimension of vectors $\mathbf{z}$ and $\mathbf{s}$. Substitute the $p(\mathbf{s})$ and $\hat{p}(\mathbf{s})$ in Eq. (2.8) into Eq. (2.6), the KL divergence for nonlinearity detection of a single Gaussian distribution, after the derivation detailed in Appendix A, can be expressed as:

$$\begin{aligned} D_{KL}(p \,\|\, \hat{p}) &= \int_{-\infty}^{\infty} p(\mathbf{s}) \log \frac{p(\mathbf{s})}{\hat{p}(\mathbf{s})} d\mathbf{s} \\ &= -\frac{k}{2} + \sum_{i=1}^{k} \log(f'(\mu_i)) - \sum_{i=1}^{k} E_{p(z_i)} \left[\log(f'(z_i)) \right] \\ &+ \frac{1}{2} \sum_{i=1}^{k} \sum_{j=1}^{k} \left(\frac{c_{ij}}{f'(\mu_i) f'(\mu_j)} E_{p(z_i, z_j)} \left[(f(z_i) - f(\mu_i))(f(z_j) - f(\mu_j)) \right] \right) \end{aligned} \tag{2.9}$$

where c_{ij} is the element in the i-th row and j-th column in the inverse of covariance matrix $\boldsymbol{\Sigma}^{-1}$ and E_p denotes the expectation with respect to the probability distribution p. In Eq. (2.9), the last two terms involve a set of 1-D and 2-D expectation integrals, which can be estimated by the unscented transform (UT) with an accuracy to the second-order Tayler expansion. The UT method might be less accurate when the covariance is large and the higher-order expansion terms are significant. However, the Eq. (2.9) is only used to determine whether a Gaussian PDF needs to be split, while it would not directly affect the uncertainty propagation accuracy. Moreover, the main computation cost of UT, i.e., the singular value decomposition, concurs with the Gaussian splitting scheme that will be discussed later. In many works regarding uncertainty propagation in neural networks, the pre-activation states are assumed to be uncorrelated, so that the covariance matrix is diagonal and the expectation integrals can be decoupled as a series of independent 1-D integrals [15][46][97]. However, such an assumption may not always hold. For example, even if a diagonal covariance $\boldsymbol{\Sigma}_{\mathbf{x}}$ is assigned to the network input $\mathbf{x}$, after transformation by the linear layer using Eq. (2.3), the

pre-activation covariance $\mathbf{\Sigma_z}$ is in general not diagonal. Therefore, non-diagonal covariance matrices are considered in this work.

For a Gaussian distribution $\mathcal{N}(\mathbf{z}|\boldsymbol{\mu}, \mathbf{\Sigma})$, a set of weighted sigma points ($2k$+1 points for a k-dimensional distribution) matching the first two moments can be generated in UT as [52]:

$$\begin{aligned} Z_0 &= \boldsymbol{\mu} \\ Z_i &= \boldsymbol{\mu} + \sqrt{(k+\lambda)}\mathbf{P}_i \quad i = 1,\ldots,k \\ Z_i &= \boldsymbol{\mu} - \sqrt{(k+\lambda)}\mathbf{P}_{i-L} \quad i = k+1,\ldots,2k \\ W_0 &= \lambda/(k+\lambda) \\ W_i &= 1/2/(k+\lambda) \quad i = 1,\ldots,2k \end{aligned} \tag{2.10}$$

where the scaling parameter λ is typically selected as λ+k=3 for a Gaussian distribution, $\mathbf{P}$ is the square root of the covariance matrix $\mathbf{\Sigma}$= $\mathbf{P}\mathbf{P}^T$ and $\mathbf{P}_i$ denotes the i-th column of $\mathbf{P}$. The sigma points match the input moments as $\sum_{i=0}^{2k+1} W_i Z_i = \boldsymbol{\mu}$ and $\sum_{i=0}^{2k+1} W_i (Z_i - \boldsymbol{\mu})(Z_i - \boldsymbol{\mu})^T = \mathbf{\Sigma}$. The square root of covariance can be obtained using the singular value decomposition (SVD), which is equivalent to the eigen-decomposition for a symmetric and positive definite covariance matrix:

$$\begin{aligned} &\mathbf{\Sigma} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^T \\ &\mathbf{V} = \begin{bmatrix} \mathbf{v}_1 & \ldots & \mathbf{v}_k \end{bmatrix} \quad \mathbf{\Lambda} = \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_k \end{bmatrix} \\ &\mathbf{P} = \begin{bmatrix} \sqrt{\lambda_1}\mathbf{v}_1 & \ldots & \sqrt{\lambda_k}\mathbf{v}_k \end{bmatrix} \quad \mathbf{\Sigma} = \mathbf{P}\mathbf{P}^T \end{aligned} \tag{2.11}$$

where $\mathbf{v}_1$, …, $\mathbf{v}_k$ are the orthonormal singular vectors (eigenvectors) and λ_1, …, λ_k are the singular values (eigenvalues). Given this decomposition, the inverse of covariance matrix can be computed as $\mathbf{\Sigma}^{-1} = \mathbf{V}\mathbf{\Lambda}^{-1}\mathbf{V}^T$ without invoking the matrix inverse routine.

Using nonlinear transformation of the sigma points, the expectation integrals in Eq. (2.9) can be estimated in vector form as:

$$E_{p(\mathbf{z})}\left[\log\left(\mathbf{f}'(\mathbf{z})\right)\right] \approx \sum_{i=0}^{2k} W_i \log\left(\mathbf{f}'(Z_i)\right) \tag{2.12}$$

and in matrix form as

$$\begin{aligned} &E_{p(\mathbf{z})}\left[\left(\mathbf{f}(\mathbf{z}) - \mathbf{f}(\boldsymbol{\mu})\right)\left(\mathbf{f}(\mathbf{z}) - \mathbf{f}(\boldsymbol{\mu})\right)^T\right] \\ &\approx \sum_{i=0}^{2k} W_i \left(\mathbf{f}(Z_i) - \mathbf{f}(\boldsymbol{\mu})\right)\left(\mathbf{f}(Z_i) - \mathbf{f}(\boldsymbol{\mu})\right)^T \end{aligned} \tag{2.13}$$

Then by summing up all the vector elements, the 1-D expectation term can be obtained as:

$$\sum_{i=1}^{k} E_{p(z_i)}\left[\log\left(f'(z_i)\right)\right] = \text{sum}\left\{E_{p(\mathbf{z})}\left[\log\left(\mathbf{f}'(\mathbf{z})\right)\right]\right\} \tag{2.14}$$

and by summing up all the matrix elements, the 2-D expectation term can be obtained as:

$$\begin{aligned}&\sum_{i=1}^{k}\sum_{j=1}^{k}\left(\frac{c_{ij}}{f'(\mu_i)\,f'(\mu_j)} E_{p(z_i,z_j)}\left[\left(f(z_i)-f(\mu_i)\right)\left(f(z_j)-f(\mu_j)\right)\right]\right)\\ &= \text{sum}\left\{E_{p(\mathbf{z})}\left[\left(\mathbf{f}(\mathbf{z})-\mathbf{f}(\boldsymbol{\mu})\right)\left(\mathbf{f}(\mathbf{z})-\mathbf{f}(\boldsymbol{\mu})\right)^T\right]\circ\boldsymbol{\Sigma}^{-1}\circ\left[\left(1/\mathbf{f}(\boldsymbol{\mu})\right)\left(1/\mathbf{f}(\boldsymbol{\mu})\right)^T\right]\right\}\end{aligned} \tag{2.15}$$

where $\circ$ denotes element-wise matrix multiplication.

The degenerate case: the above analysis is based on the assumption that the covariance matrix is non-singular so that it is invertible and has k independent eigenvectors. Nevertheless, singular covariance matrices could occur in neural networks. Assuming the covariance matrix $\boldsymbol{\Sigma}$ has a rank of r ($r<k$), then the PDF is only supported on a r-dimensional subspace. In such a degenerate case, the $k/2$ term in Eq. (2.9) should be replaced with $r/2$ since it comes from the differential entropy and the entropy also degenerates. The inverse of covariance matrix can be obtained using the truncated pseudoinverse approximation $\boldsymbol{\Sigma}^{-1} \approx \sum_{i=1}^{r} \mathbf{v}_i \mathbf{v}_i^T / \lambda_i$, where $\mathbf{v}_i$ are the singular vectors and λ_i are the non-zero singular values [207]. In this way, the KL divergence in Eq. (2.9) could be estimated for a degenerated distribution on the r-dimensional subspace. A primary cause of singular covariance in a neural network is because the number of inputs n is typically less than the number of hidden nodes k. Even if the input $\mathbf{x}$ has a non-singular covariance $\boldsymbol{\Sigma}_\mathbf{x}$, after the transformation in Eq. (2.3), the rank r of the pre-activation covariance $\boldsymbol{\Sigma}_\mathbf{z}$ is at most n, i.e., $r \leq n < k$. In such a scenario, the SVD can be performed on the low-dimensional covariance $\boldsymbol{\Sigma}_\mathbf{x}$ instead of $\boldsymbol{\Sigma}_\mathbf{z}$ for better computation efficiency.

Now the KL-divergence criterion to assess the level of nonlinearity has been established. For a pre-activation Gaussian PDF, if its KL divergence in Eq. (2.9) is larger than a threshold, it will be split into narrower ones as described in the next subsection.

2.2.2 Gaussian Splitting Scheme

To develop the Gaussian splitting scheme, the univariate Gaussian distribution is studied first and then the scheme is extended to multivariate cases. A standard univariate Gaussian distribution $p(z) = \mathcal{N}(z|0,1)$ could be approximated by a Gaussian mixture with P components as:

$$\tilde{p}(z) = \sum_{i=1}^{P} \tilde{w}_i \mathcal{N}\left(z \mid \tilde{\mu}_i, \tilde{\sigma}^2\right) \tag{2.16}$$

where $\tilde{w}_i$, $\tilde{\mu}_i$ are the weights and means of mixture components and all the components share the same standard deviation $\tilde{\sigma}$. These Gaussian mixture parameters can be found by minimizing the KL divergence between the standard Gaussian distribution and the approximating mixture as:

$$\begin{aligned} &\min_{\tilde{w}_i, \tilde{\mu}_i, \tilde{\sigma}} \quad \tilde{D}_{KL}\left(p(z) \| \hat{p}(z)\right) + \alpha P \tilde{\sigma} \\ &\text{subject to } \sum_{i=1}^{P} \tilde{w}_i = 1 \quad 0 < \tilde{\sigma} < 1 \end{aligned} \tag{2.17}$$

where α is a scaling factor of the penalty term to enforce that with more components P, a more aggressive splitting with smaller standard deviation should be adopted to reduce the magnitude of uncertainty carried by each component. By setting $\alpha = 0.001$ and using odd numbers of P with a center component always located at the origin, the optimization in Eq. (2.17) was solved using the genetic algorithm. The optimization results for three, five and seven components are summarized in Table 2-2 and Figure 2-2.

Table 2-2 The splitting schemes for the standard Gaussian distribution

Number of components	$\tilde{w}_i$	$\tilde{\mu}_i$	$\tilde{\sigma}$	$\tilde{D}_{KL}$
3	0.6364 0.1818	0 ±1.0579	0.7687 0.7687	1.77×10^{-4}
5	0.4444 0.2455 0.0323	0 ±0.9332 ±1.9776	0.5654 0.5654 0.5654	2.34×10^{-4}
7	0.3048 0.2410 0.0948 0.0118	0 ±0.7056 ±1.4992 ±2.4601	0.4389 0.4389 0.4389 0.4389	3.56×10^{-4}

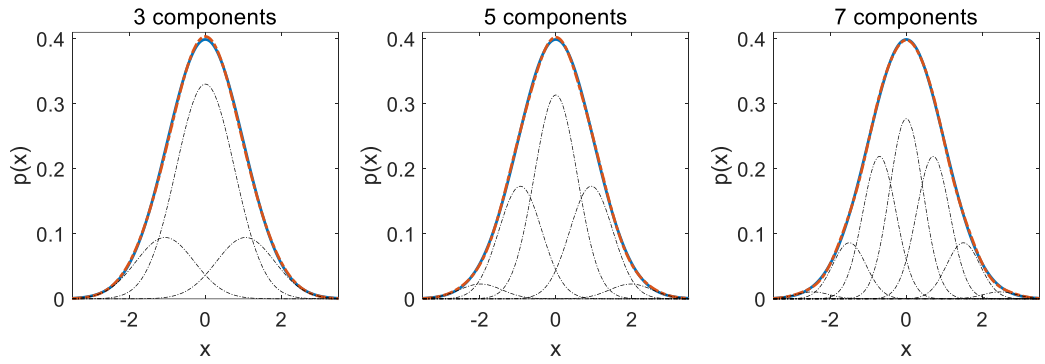


Figure 2-2 The splitting schemes for the standard Gaussian distribution

Solid bold: true distribution, dashed bold: Gaussian mixture approximation, dashed thin: components

For a multivariate Gaussian distribution $p(\mathbf{z}) = \mathcal{N}(\mathbf{z}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$, given that its covariance matrix has been decomposed into singular vectors as in Eq. (2.11), a univariate splitting scheme can be extended to split this multivariate distribution along a certain singular vector:

$$\tilde{\boldsymbol{\mu}} = \boldsymbol{\mu} + \tilde{\mu}_p \sqrt{\lambda_j}\, \mathbf{v}_j \quad \tilde{\boldsymbol{\Sigma}} = \mathbf{V}\tilde{\boldsymbol{\Lambda}}\mathbf{V} \quad \tilde{\boldsymbol{\Lambda}} = diag\left\{\lambda_1 \quad \ldots \quad \tilde{\sigma}^2\lambda_j \quad \ldots \quad \lambda_r\right\} \tag{2.18}$$

where λ_i and $\mathbf{v}_i$ are the singular values and singular vectors, r is the rank of $\boldsymbol{\Sigma}$, and j is the index of the principal axis selected for splitting. In this work, the singular vector corresponding to the largest singular value will be selected for splitting since this is the direction with the most significant magnitude of uncertainty.

To show that the KL divergence of splitting a multivariate Gaussian distribution equals the KL divergence of the univariate splitting scheme adopted, first consider a standard k-dimensional Gaussian distribution with zero mean vector and identity covariance matrix $p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\mathbf{0}, \mathbf{I}_k)$. Since the k variables are uncorrelated, the multivariate PDF is the product of k univariate PDFs, which means a splitting on one dimension doesn't affect the KL-divergence integrations on other dimensions. Then by a change of variable, the standard Gaussian distribution can be transformed to an arbitrary Gaussian distribution without altering the value of KL divergence integration. The invariance of KL divergence under linear coordinate transformations makes the quality of Gaussian distribution splitting consistent regardless of the mean and covariance.

2.2.3 Gaussian Mixture Propagation

If the pre-activation distribution in a neural network is characterized as a Gaussian mixture, then the nonlinearity detection and Gaussian splitting methods discussed above can be applied to each mixture component. For a Gaussian component, its KL-divergence measure of nonlinearity in Eq. (2.9) is scaled by its weight w and then compared with a threshold th_D. If the divergence is larger than the threshold, i.e., $w\hat{D}_{KL}\left(p \,\|\, \hat{p}\right) > th_D$, a scheme in Table 2-2 will be used to split this component. This comparison is made by considering the weight so that those components with minor contributions to the overall PDF will be less likely to get involved in splitting. The new components from splitting with weights $w\tilde{w}_p$ will be added to the mixture to replace the original one. The split components share the same singular vectors and singular values inherited from the original component except that the singular value selected for splitting has been scaled down. Hence, the SVD doesn't need to be repeated for any of the new split components.

The above process will be repeated until the KL divergences of all the components are less than the threshold. Then all the components can be easily propagated through activation function using the unscented transform (UT) with minor computation cost since the SVDs have already been available for each component. For a pre-activation component $p^i\left(\mathbf{z}\right) = w_{\mathbf{z}}^i \mathcal{N}\left(\mathbf{z} \,|\, \boldsymbol{\mu}_{\mathbf{z}}^i, \boldsymbol{\Sigma}_{\mathbf{z}}^i\right)$, the optimal post-activation Gaussian approximation that matches the first two moments can be estimated using the UT as:

$$
\begin{aligned}
& p^i\left(\mathbf{s}\right) = w_{\mathbf{s}}^i \mathcal{N}\left(\mathbf{s} \,|\, \boldsymbol{\mu}_{\mathbf{s}}^i, \boldsymbol{\Sigma}_{\mathbf{s}}^i\right) \\
& w_{\mathbf{s}}^i = w_{\mathbf{z}}^i \\
& \boldsymbol{\mu}_{\mathbf{s}}^i = E_{p^i(\mathbf{z})}\left[\mathbf{f}\left(\mathbf{z}\right)\right] \approx \sum\nolimits_{j=0}^{2k} W_j^i \mathbf{f}\left(Z_j^i\right) \\
& \boldsymbol{\Sigma}_{\mathbf{s}}^i = E_{p^i(\mathbf{z})}\left[\left(\mathbf{f}\left(\mathbf{z}\right) - \boldsymbol{\mu}_{\mathbf{s}}^i\right)\left(\mathbf{f}\left(\mathbf{z}\right) - \boldsymbol{\mu}_{\mathbf{s}}^i\right)^T\right] \\
& \qquad \approx \sum\nolimits_{j=0}^{2k} W_j^i \left(\mathbf{f}\left(Z_j^i\right) - \boldsymbol{\mu}_{\mathbf{s}}^i\right)\left(\mathbf{f}\left(Z_j^i\right) - \boldsymbol{\mu}_{\mathbf{s}}^i\right)^T
\end{aligned}
\tag{2.19}
$$

where $\mathbf{z}$ and $\mathbf{s}$ are the pre-activation and post-activation state vectors, i is the index of the i-th component, and Z_j^i and W_j^i are the sigma points and weights. The algorithm of Gaussian mixture propagation through a nonlinear activation layer is summarized below in Algorithm I.

Algorithm I: adaptive uncertainty propagation in a nonlinear activation layer

Input: the *M*-component pre-activation Gaussian mixture PDF $p(\mathbf{z}) = \sum_{i=1}^{M} w_{\mathbf{z}}^{i} \mathcal{N}\left(\mathbf{z} \mid \boldsymbol{\mu}_{\mathbf{z}}^{i}, \boldsymbol{\Sigma}_{\mathbf{z}}^{i}\right)$ and the nonlinearity threshold th_D

Nonlinearity Evaluation and Splitting

(1) For a component *i*, perform SVD its covariance matrix
(2) Use UT to calculate Eq. (2.12) ~ (2.15)
(3) Substitute Eq. (2.14) and (2.15) into Eq. (2.9) to get the KL divergence $D_{KL}\left(p^i \parallel \hat{p}^i\right)$
(4) If $w_{\mathbf{z}}^{i} D_{KL}\left(p^i \parallel \hat{p}^i\right) > th_D$, split this component using Eq. (2.18) with a scheme from Table 2-2. All the components from splitting have the same covariance.
(5) Repeat (1)-(4) for all the *M* components
(6) Repeat (2)-(5) for all the new components from splitting until there is no new splitting

Gaussian Mixture Propagation

(7) Use UT in Eq. (2.19) to estimate the post-activation PDF for each of the *N* components after splitting (*N*>*M*)

Output: the post-activation PDF as an *N*-component mixture $p(\mathbf{s}) = \sum_{i=1}^{N} w_{\mathbf{s}}^{i} \mathcal{N}\left(\mathbf{z} \mid \boldsymbol{\mu}_{\mathbf{s}}^{i}, \boldsymbol{\Sigma}_{\mathbf{s}}^{i}\right)$

2.2.4 Gaussian Mixture Reduction

A problem that may arise from the splitting scheme in uncertainty propagation is that the number of components will keep increasing though the actual probability density function is not going to be infinitely complicated. Therefore, a Gaussian mixture reduction step is often used to merge the components to maintain computational efficiency. There have been various Gaussian mixture reduction methods in the literature, among which the Kullback-Leibler approach in [208] is found to better preserve the PDF shapes and hence is adopted in this chapter.

A Gaussian mixture reduction algorithm successively selects a pair of components that are close together and similar in shape and merges them into one component. Assuming that the two components are parameterized by $p_i \sim \{w_i, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\}$ and $p_j \sim \{w_j, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j\}$, the moment-preserving merging of these two components $p_{ij} \sim \{w_{ij}, \boldsymbol{\mu}_{ij}, \boldsymbol{\Sigma}_{ij}\}$ is given as:

$$
\begin{aligned}
w_{ij} &= w_i + w_j \\
\boldsymbol{\mu}_{ij} &= \frac{w_i \boldsymbol{\mu}_i + w_j \boldsymbol{\mu}_j}{w_i + w_j} \\
\boldsymbol{\Sigma}_{ij} &= \frac{w_i \boldsymbol{\Sigma}_i + w_j \boldsymbol{\Sigma}_j}{w_i + w_j} + \frac{w_i w_j \left(\boldsymbol{\mu}_i - \boldsymbol{\mu}_j\right)\left(\boldsymbol{\mu}_i - \boldsymbol{\mu}_j\right)^T}{\left(w_i + w_j\right)^2}
\end{aligned}
\tag{2.20}
$$

where p_{ij} has the same weight, mean and covariance as the mixture of p_i and p_j. It may be intuitive to select the pair of components with the least KL divergence for merging so that the PDF misrepresentation can be minimized. Although this divergence does not have a closed-form solution, an easy-to-compute upper bound of it has been derived by Runnalls [208] as:

$$B_{i,j} = \frac{1}{2}\left[\left(w_i + w_j\right)\log\left|\boldsymbol{\Sigma}_{ij}\right| - w_i \log\left|\boldsymbol{\Sigma}_i\right| - w_j \log\left|\boldsymbol{\Sigma}_j\right|\right] \tag{2.21}$$

This upper bound is a conservative estimation of the PDF distortion caused by merging two components. In the proposed uncertainty propagation scheme, a component p_i will be paired with each of the rest components and the one that yields the lowest bound $B_{i,j}$ will be merged with p_i, if the bound is also lower than a threshold th_B. The threshold is imposed to prevent a large mis-representation from being introduced by the reduction. Given the symmetricity ($B_{i,j} = B_{j,i}$), an N-component mixture needs a total of $N(N-1)/2$ bound evaluations in one reduction procedure, and this procedure can be repeated until no upper bound evaluation drops below the threshold.

2.2.5 Preliminary Tests on Activation Functions

Before applying the established uncertainty propagation scheme onto neural networks, it was first tested on some of the nonlinear activation functions to examine its fidelity in approximating non-Gaussian post-activation PDFs. Figure 2-3 shows the transformation of a Gaussian distribution by the hyperbolic tangent function (tanh). The true distribution appears to be highly skewed due to the nonlinearity of tanh function and the Gaussian mixture obtained using the proposed method approximated this distribution accurately with a KL divergence of 0.0020. Figure 2-4 (a) presents the transformation of a Gaussian distribution by ReLU function, which is the most commonly used activation function in deep neural networks. On the negative half axis of ReLU, a bias of 0.001 was added to the derivative to prevent the zero derivatives to make the logarithm terms in Eq. (2.9) become infinity. As can be seen, the proposed method successfully predicted the post-activation PDF, including both the Dirac delta distribution part lumped at 0 and the truncated Gaussian part on the positive half axis. The transformation of the same Gaussian distribution by a leaky ReLU function is also shown in Figure 2-4 (b), for which the proposed method attained a level of fidelity similar to the ReLU transformation.

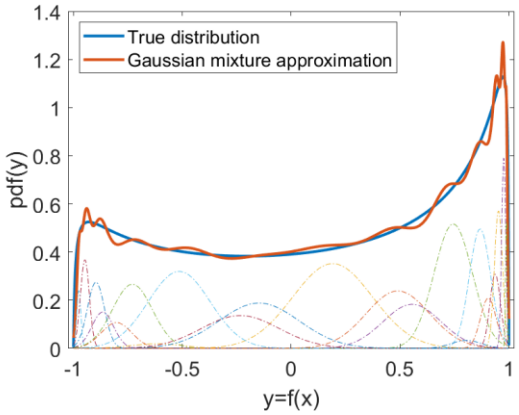
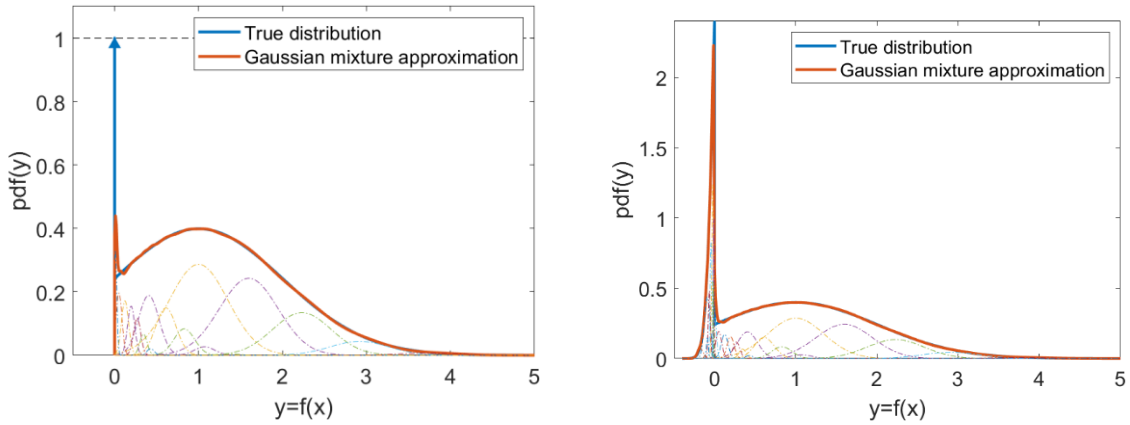


Figure 2-3 Approximation of a univariate skewed distribution from a tanh function
Input Gaussian: μ = 0.25, σ2 =1; Threshold: thD = 0.01, thB = 0.001; 7-component splitting
Results: number of components: 29, KL divergence: 0.0020, computation time: 1.6ms



(a) ReLU (b) Leaky ReLU (c=0.1)

Figure 2-4 Approximation of multimodal distributions from a ReLU and a Leaky ReLU function
Input Gaussian: μ = 1, σ2 = 1; Threshold: thD = 0.001, thB = 0.001; 7-component splitting
Results: (a) ReLU: number of components: 36, KL divergence: 0.0039, computation time: 1.7ms
(b) Leaky ReLU: number of components: 27, KL divergence: 0.0048, computation time: 1.6ms

Then in Figure 2-5, a 2-dimensional Gaussian distribution was propagated through the logistic function, and the proposed method successfully constructed a Gaussian mixture that approximates the distorted post-activation PDF with a KL divergence of 0.0037. In addition, the computation time was at millisecond level in all the tests. Therefore, both the fidelity and computational efficiency of the proposed Gaussian mixture scheme have been verified.

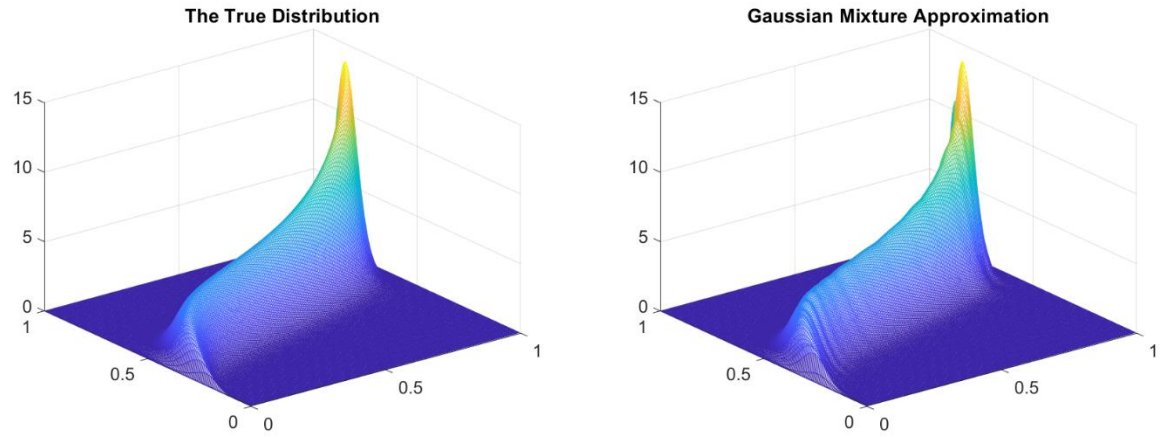


Figure 2-5 Approximation of a bivariate distorted distribution from a logistic function

Input Gaussian: μ = [0.5, 0.5]T, σ2 = [2.72, 0.80; 0.80, 0.32];

Threshold: thD = 0.001, thB = 0.001; 7-component splitting

Results: number of components: 77, KL divergence: 0.0025, computation time: 4ms

2.3 Application Examples

Upon the success in preliminary tests, the proposed scheme was applied to a series of nonlinear uncertainty propagation problems to further prove its effectiveness for propagating uncertainties through time in recurrent networks and layers in multilayer networks

2.3.1 Example I: A Noise-driven Nonlinear Damping Oscillator

The first application example is a noise-driven nonlinear damping oscillator. The equation of motion of this damping system is given as:

$$\ddot{x} + \beta\dot{x} + x + \alpha\left(x^2 + \dot{x}^2\right)\dot{x} = gG(t) \tag{2.22}$$

where $\beta = -0.5$, $\alpha = 0.5$, $g^2 = 0.75$ were selected for simulation and a Gaussian white noise $G(t)$ with a variance of 2 was added. To implement the proposed method, the dynamics of this system need to be approximated by a neural network. By using a sampling interval of 0.05 seconds and a simulation time of 10 seconds, 1000 data samples were collected from 5 simulations using Eq. (2.22). Then a recurrent neural network (a two-layer network with output feedback) with 5 hidden nodes was trained to approximate the system dynamics in the following form:

$$\begin{aligned}\mathbf{x}_k &= \mathbf{W}_2\mathbf{f}\left(\mathbf{W}_1\left[\mathbf{x}_{k-1}{}^T, G_{k-1}\right]^T + \mathbf{b}_1\right) + \mathbf{b}_2 \\ \mathbf{x}_k &= \left[x_k \quad \dot{x}_k\right]^T\end{aligned} \tag{2.23}$$

where x_k, $\dot{x}_k$ and G_k are the oscillator position, velocity, and noise at time step k, respectively.

Such an oscillator has a stationary PDF that can be expressed as [209]:

$$p\left(x,\dot{x}\right) = C\exp\left\{-\frac{1}{2g^2}\left[\beta\left(x^2+\dot{x}^2\right)+\alpha\left(x^2+\dot{x}^2\right)^2\right]\right\} \tag{2.24}$$

where C is a normalization constant. The proposed uncertainty propagation method was applied to the trained recurrent network to predict this stationary PDF. The network structure parameters and uncertainty propagation parameters are summarized in Table 2-3. The initial uncertainty was given as a Gaussian distribution with the mean at the origin and a diagonal covariance with standard deviations of 0.1 for both the position and velocity.

Table 2-3 Network and uncertainty propagation parameters in the application examples

	Example I	*Example II*	*Example III*	*Example IV*	*Example V*
Input size	3 (1 external+2 feedback)	4 (all feedback)	16 (4 external+12 feedback)	4 (all external)	784 (all external)
Output size	2	4	12	1	10
Number of hidden layers	1	1	1	3	4
Number of hidden nodes	5	20	50	20 ea. layer	200 ea. layer
Activation function	Logistic	Logistic	Tanh	Leaky ReLU	ReLU
Nonlinearity threshold th_D	10^{-4}	10^{-4}	10^{-3}	10^{-4}	10^{-2}
Splitting scheme	5	5	5	5	5
Reduction threshold th_B	10^{-3}	N/A	10^{-2}	10^{-3}	10^{-1}

After propagating uncertainty for 10 seconds, the PDF predicted by the proposed method and the theoretical stationary PDF in Eq. (2.24) are compared in Figure 2-6. It can be seen that the predicted Gaussian mixture correctly reconstructed the "volcano" shaped stationary PDF. In addition, a Monte Carlo (MC) simulation with 10 million samples was also performed and the PDF

at t =10 second was extracted from the output histogram. The result predicted using the proposed method, the MC result and the result in [57] using a quadratic programming approach were compared in Table 2-4. As can be seen, the proposed method achieved the lowest KL divergence and absolute error with respect to the true theoretical PDF, while its computation time was only 4% of the MC simulation time. Compared with the result in [57], the proposed method used many fewer Gaussian mixture components (112 vs. 1000), while the absolute error was reduced significantly by 75%. Since the proposed method outperformed the MC simulation when assessed using the theoretical solution as the ground truth, it may be used as a reliable and efficient PDF prediction scheme for the nonlinear damping systems without theoretical solutions, given that their nonlinear damping dynamics can be approximated by neural networks.

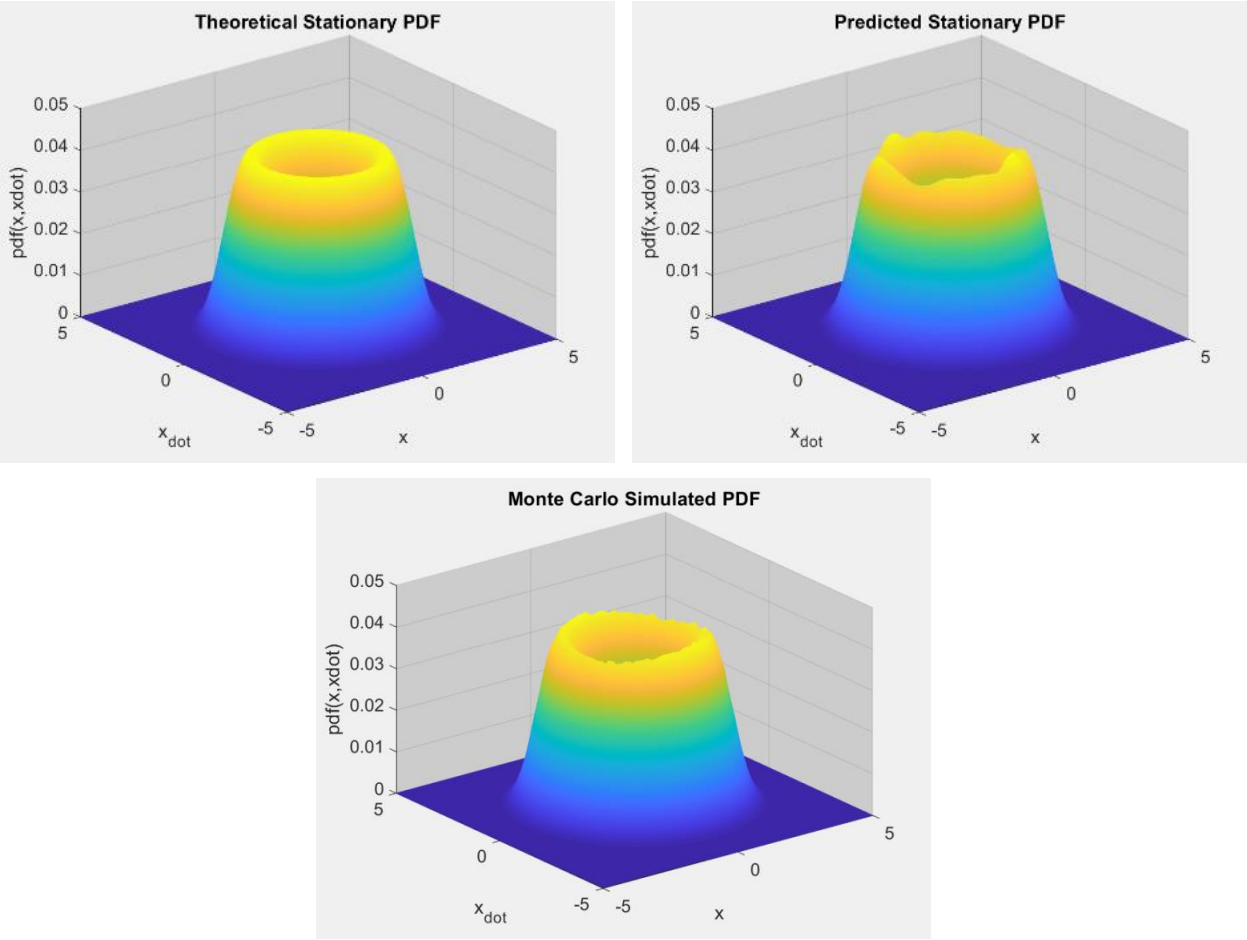


Figure 2-6 Comparison of theoretical, MC and predicted PDFs for Example I

Table 2-4 Comparison of the results for uncertainty propagation Example I

	Number of components	*KL divergence to the true distribution*	*Absolute error to the true distribution*	*Computation time*
Proposed Method	112	0.0033	0.0477	2.1 sec
Monte Carlo	10 million (10^7)	0.0057	0.0760	~ 52 sec
Method in [57]	1000	N/A	0.19	N/A

2.3.2 Example II: Low-earth-orbit Uncertainty Tracking

The second example is the uncertainty propagation in a low-earth-orbit tracking problem from [61]. The equations of motion for an object's position and velocity in the orbit plane are:

$$\mathbf{x}(t)=\begin{bmatrix} x \\ y \\ \dot{x} \\ \dot{y} \end{bmatrix} \quad \dot{\mathbf{x}}(t)=\begin{bmatrix} \dot{x} \\ \dot{y} \\ -\mu x\left(x^2+y^2\right)^{-\frac{3}{2}}-\frac{1}{2}\rho(h)\beta v_r v_{r,x} \\ -\mu y\left(x^2+y^2\right)^{-\frac{3}{2}}-\frac{1}{2}\rho(h)\beta v_r v_{r,y} \end{bmatrix} \tag{2.25}$$

where μ is the gravitational constant of the earth, $v_{r,x}=\dot{x}+\omega y$ and $v_{r,y}=\dot{y}-\omega x$ are the object's relative velocity with respect to the atmosphere, $v_r=\sqrt{{v_{r,x}}^2+{v_{r,y}}^2}$ is the resultant velocity, ω is the angular velocity of the earth, β is the object's ballistic coefficient (β = 1.4 in this example), ρ is the atmospheric density $\rho=\rho_0\exp\left[-\left(h-h_0\right)/h_s\right]$ with ρ_0 = 3.614×10^{-13} kg/m^3, h_0 = 700 km and h_s = 88.667 km, $h=\sqrt{x^2+y^2}-R_e$ is the altitude of the orbit and R_e is the radius of the earth. The initial state uncertainty was assumed to be Gaussian with the mean on a circular orbit with an altitude of 225 km starting from y = 0 and a diagonal covariance with standard deviations of 1.3 km in x position, 0.5 km in y position, 2.5 m/s in x velocity, and 5 m/s in y velocity.

A neural network with 20 hidden nodes was trained to approximate the orbit dynamics, i.e., the $\mathbf{x}$ to $\dot{\mathbf{x}}$ mapping in Eq. (2.25), by collecting 3600 training samples from simulations of Eq. (2.25). Then the neural network was implemented in a recurrent form with output feedback, and a sampling time dt = 0.1 seconds was used to discretize the system using the Euler method:

$$\begin{aligned} \dot{\mathbf{x}}_{k-1} &= \mathbf{W}_2\mathbf{f}\left(\mathbf{W}_1\mathbf{x}_{k-1}+\mathbf{b}_1\right)+\mathbf{b}_2 \\ \mathbf{x}_k &= \mathbf{x}_{k-1}+\dot{\mathbf{x}}_{k-1}dt \end{aligned} \tag{2.26}$$

The initial uncertainty was fed into the recurrent network for propagation using the proposed method. The network structure and uncertainty propagation parameters are summarized in Table 2-3. Since there was no extra uncertainty (as the Gaussian white noise in Example I) added in each time step, the Gaussian mixture reduction was turned off. A Monte Carlo simulation of 10^5 samples was performed to estimate the ground truth distributions since there was no analytical solution for this problem. The PDF contours predicted by the proposed method after one period of nominal orbit (about 1.5 hours) are compared with MC samples in Figure 2-7 and the predicted position PDF is compared with the histogram of MC samples in Figure 2-8. As can be seen, the Gaussian mixture obtained using the proposed method correctly predicted the "banana" shaped PDF for both position and velocity. The KL divergence of the predicted PDF with respect to the PDF extracted from the MC histogram was 0.0114 for position and 0.0139 for velocity. The computation time of the proposed method to propagate uncertainty for one orbit period was about 40 seconds on a single thread CPU, while the MC simulation took about 2200 seconds on the same platform. The test was then extended to two orbit periods (about 3 hours), and the predicted PDF of position is compared with the MC histogram in Figure 2-9. The proposed method predicted the distribution of position correctly in this long-term forecasting with the KL divergence increased moderately to 0.0596. Compared with the Gaussian mixture results achieved in [61], the proposed method used fewer components (25 vs. 150 at one period and 125 vs. 350 at two periods) without compromising fidelity, and thus it is expected to be more computationally efficient.

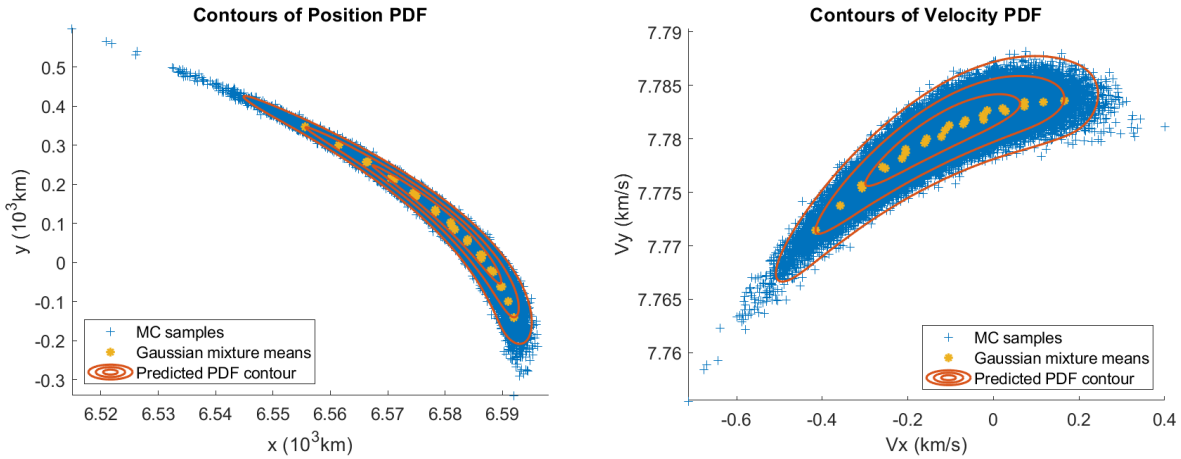


Figure 2-7 Predicted PDF contours compared with Monte Carlo samples at one orbit period
The orange contours are for probabilities of 68.3%, 95.4%, and 99.7%

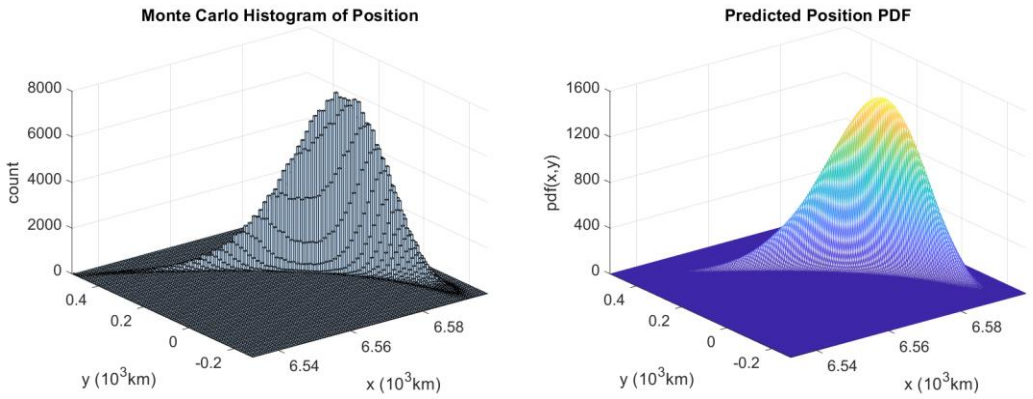


Figure 2-8 Position PDF compared with the Monte Carlo histogram at one orbit period

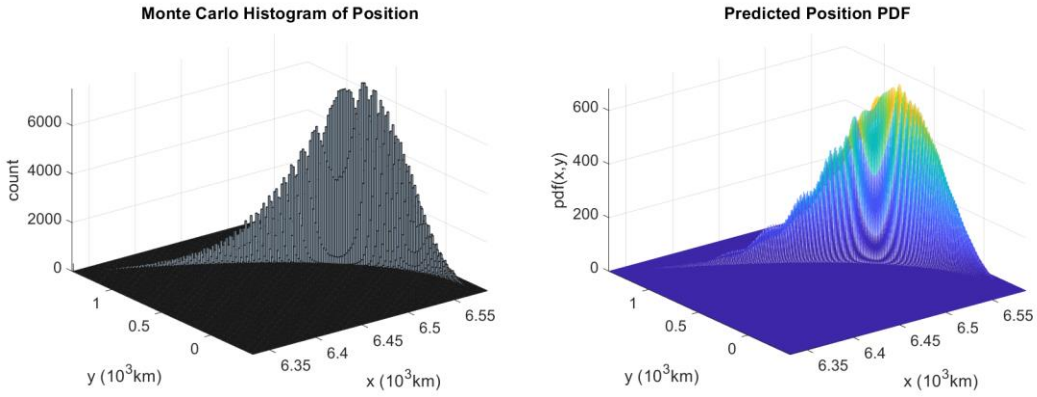


Figure 2-9 Position PDF compared with the Monte Carlo histogram at two orbit period

2.3.3 Example III: Path Uncertainty Prediction for a Quadrotor Drone

The third example is the path uncertainty propagation for a quadrotor drone. Estimating the trajectory uncertainty is important for path planning and collision avoidance of unmanned aerial vehicles [210]. The drone system considered in this work has 12 state variables and 4 inputs:

$$\begin{aligned}\mathbf{x} &= \begin{bmatrix} x & y & z & \dot{x} & \dot{y} & \dot{z} & \phi & \theta & \psi & p & q & r \end{bmatrix}^T \\ \mathbf{u} &= \begin{bmatrix} \omega_1 & \omega_2 & \omega_3 & \omega_4 \end{bmatrix}^T \end{aligned} \tag{2.27}$$

where the first 6 states are x, y, z locations of the drone in the inertial frame and their velocities, φ, ψ, θ are the pitch (rotation around x axis), roll (around y axis) and yaw (around z axis) angles of the drone in inertial frame and p, q, r are the angular velocities in the body frame. The four inputs in **u** are angular velocities of the four propellers, which are used to control the total thrust force

and torques around pitch, roll and yaw angles. The equation of motion of this drone system is too complicated to be detailed in this section. A full description of the quadrotor drone dynamics can be found in Appendix B.

In this example, 10000 training data samples were collected from the simulations using the equations of motion and drone parameters given in [211], and a neural network with output feedback and 50 hidden nodes was trained to approximate this dynamic system. At the initial state, the drone was assumed to take off from rest at the origin of the inertial frame. The pitch angle had a Gaussian distribution with the mean at 1° while the roll angle had the mean at −1°, and both of their standard deviations were 0.05°. All the other positions, angles and velocities were taken to be zero at time t = 0 seconds without uncertainty. The angular velocities of four propellers were all maintained at 650 rad/s by the controller. A zero-mean Gaussian disturbance with 32.5 rad/s standard deviation was added to each propeller velocity as input uncertainty. The proposed method was applied to the network to predict the uncertainty of the drone's motion. The uncertainty propagation parameters are summarized in Table 2-3. In Figure 2-10, the predicted PDF contours of the drone's position in the X-Y plane at t = 10 seconds are compared with a Monte Carlo simulation of 1 million samples. And the predicted X-Z position PDF is compared with the histogram of MC samples in Figure 2-11. It can be seen that the PDFs obtained by the proposed method correctly predicted the distribution of the drone's location around its nominal path. The KL divergence of the predicted marginal PDF with respect to the PDF extracted from the MC histogram was 0.0319 for the X-Y position and 0.0957 for the X-Z position. Therefore, the proposed method can be possibly used for trajectory uncertainty prediction in path planning of quadrotor drones, especially in the scenarios when the environment is so complicated that the physics-based model is not sufficient to predict the drone's behavior and a neural network model trained from sensor measurement data is preferably used for system identification.

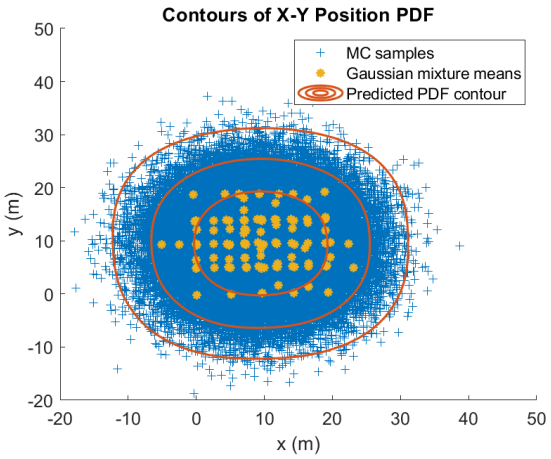


Figure 2-10 Predicted X-Y position PDF contours compared with Monte Carlo samples The orange contours are for probabilities of 68.3%, 95.4% and 99.7%

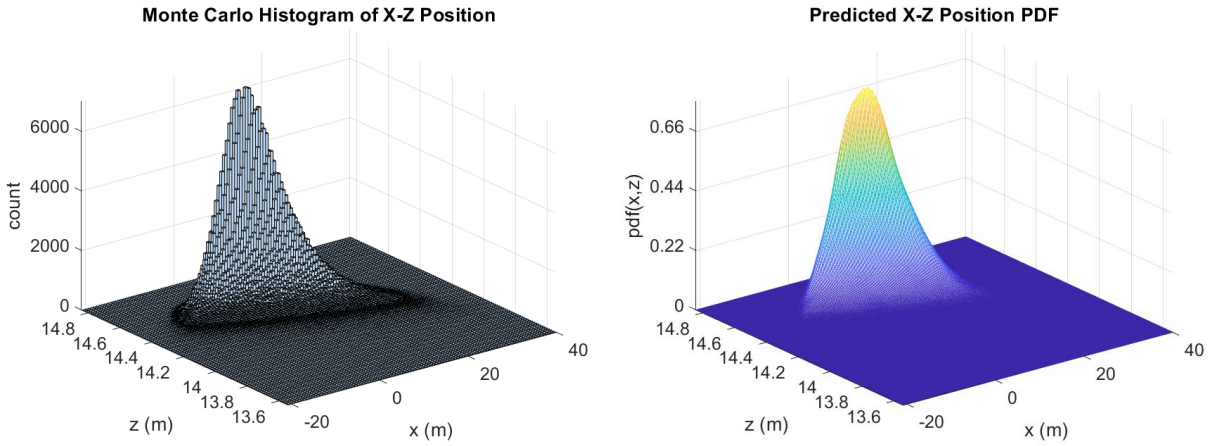


Figure 2-11 X-Z position PDF compared with the Monte Carlo histogram

2.3.4 Example IV: Power Output Prediction of a Power Plant

The fourth example is a data-driven example to predict the power generation in a combined cycle power plant. The power plant consists of 2 gas turbines and 1 steam turbine and is designed with a nominal generating capacity of 480 megawatts (MW). Four features measured by sensors are used as input variables, which include the ambient temperature, atmospheric pressure, relative humidity and vacuum. The full-load electrical power output is the target variable to predict, which ranges from 420.26 to 495.76 MW. The dataset contains 9568 data points of hourly averaged measurements of input features and power output, collected over 6 years (2006–2011). The

detailed analysis of the data can be found in [212] and the dataset can be downloaded from the UCI website [213]. In this example, to test the proposed method's capacity of layer-wise uncertainty propagation in a deeper neural network, a feedforward neural network with three hidden layers and leaky ReLU activation function was designed to fit a regression model from the dataset. The trained model achieved a root mean square error of 5.2 MW, which was on the same level of performance as the machine learning models reported in [15] and [212].

To implement the uncertainty propagation on this multilayer network model, the dataset was sorted by the output power magnitude. Then for each data instance, the uncertainty associated with input features was quantified as a Gaussian distribution with the mean at the instance feature values and the covariance calculated from 7 adjacent instances, including this instance, 3 instances below it and 3 above it. The quantified input uncertainty was propagated through the multilayer network and the output uncertainty was predicted as a Gaussian mixture. The 99% confidence intervals of the predicted Gaussian mixture are compared with the true output powers in Figure 2-12 for a random draw of 100 instances. As can be seen, the proposed method correctly predicted the intervals that enclosed true output powers by propagating the virtually reconstructed input uncertainty. Next, the proposed method was tested to propagate the probability distribution of the entire dataset instead of individual instances. A Gaussian mixture model with 6 components was fitted to characterize the joint PDF of four input features and this Gaussian mixture was fed into the trained neural network for propagation. The fitted marginal PDF of ambient temperature, which is the input feature most correlated to output power, is compared with the histogram of data in Figure 2-13 (a) to present the accuracy of the fitted input Gaussian mixture. The output power PDF predicted by propagating the input Gaussian mixture is compared with the histogram PDF estimated from the entire dataset in Figure 2-13 (b). It can be seen that the predicted power PDF matched all the modes and magnitudes of the probability in the histogram. Therefore, given that the input-output mapping has been learned by a neural network and the distribution of input features are represented as a Gaussian distribution or a Gaussian mixture, the proposed method could reliably predict the distribution of electrical power output generated by the power plant.

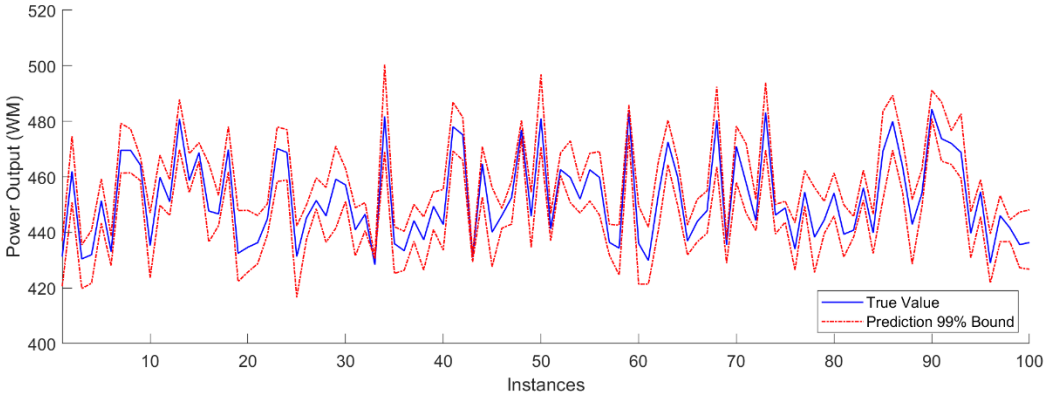
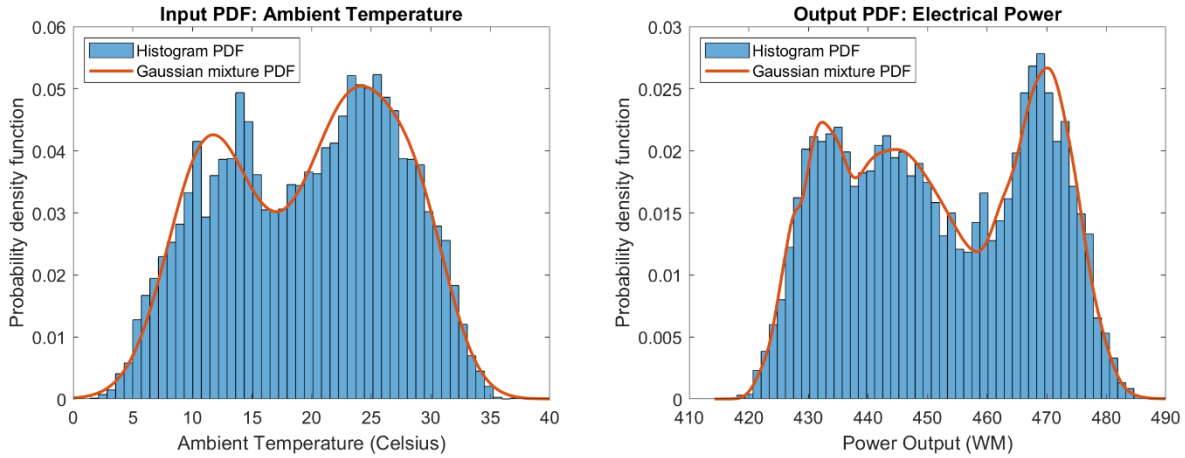


Figure 2-12 The uncertainty bounds predicted for output power



(a) (b)

Figure 2-13 Comparison of the Gaussian mixture PDF and histogram PDF from data

(a) PDFs for an input feature, the Gaussian mixture was fitted from data. (b) PDFs for the output power, the Gaussian mixture was predicted by the proposed method.

The histogram PDFs were obtained by normalizing the histogram counts.

2.3.5 Example V: MNIST Dataset Classification

The last example is on the classification of the famous MNIST dataset for handwritten digits recognition [214]. To achieve robust results, the 60000 training data samples (28×28 pixel images) were augmented by four times using the elastic distortion method in [215] to increase the total size of training set to 3×10^5. A feedforward neural network with four hidden layers and 200 ReLU hidden nodes in each layer was designed. The class labels of original and distorted images were encoded as a one-hot regression target (10 channels, with -0.1 for all the incorrect classes and 0.9

for the correct class) [51], so that the network can be trained on the mean square error (MSE) loss without a softmax layer. The training via the stochastic gradient descent (SGD) algorithm achieved an error rate of 0.86% on the 10000 test samples, which implies that the network's size and performance were both close to the multilayer perceptron models in [215].

To test uncertainty propagation on the multilayer neural network, a bimodal noise, which is characterized as a Gaussian mixture of two equally weighted members with ±5 mean vectors and diagonal covariance matrices with standard deviations of 10, was added to the pixels of images. The proposed method was implemented to propagate this input uncertainty using the parameters in Table 2-3. For comparison, two UT-based methods were also applied: one is UT for the entire neural network that only generates sigma points once at the input for each of the two noise components and the other is layer-wise UT that performs SVD and generates sigma point before each ReLU layer. The accuracies of the proposed method and the two UT methods are evaluated by comparing their predicted distributions with the ground truth estimated using a Monte Carlo simulation of 10000 samples for each image, as summarized in Table 2-5. All the evaluations were on the output channels of the correct classes (i.e., the ideal output should be 0.9 according to the one-hot encoding) and averaged over the 10000 test dataset images. As can be seen, the proposed method with adaptive splitting achieved a noticeable improvement over the layer-wise UT method that straightly propagates the two input noise components, though the computation cost also increased moderately. The Entire-NN UT method was the least accurate as the sigma points produced at the input will lose accuracy in matching the moments through subsequent layers, while its computation time was the longest owing to the costly SVD on the 784-dimensional input. Then in Figure 2-14, the predicted output PDFs for a test image (a handwritten digit 9) are compared, in which the proposed method outperformed the layer-wise UT in matching the MC histogram. Therefore, the proposed method proves to be effective in propagating uncertainty through the multilayer structures of neural networks, though the high dimensional SVD might still be a computational bottleneck for real-time implementation on very large networks.

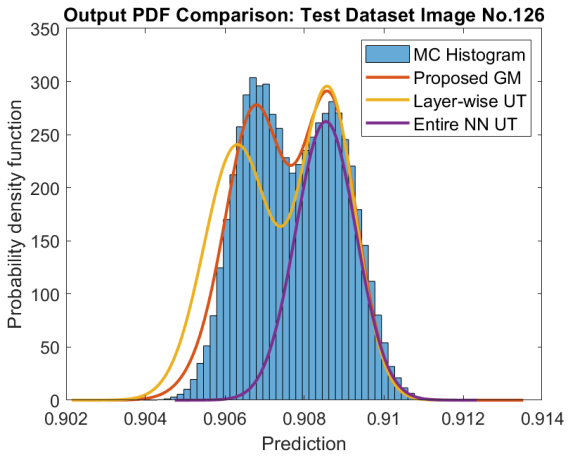


Figure 2-14 Comparison of predicted output PDFs for a MNIST image

The PDFs are marginalized on the one-hot channel for digit 9

Table 2-5 Comparison of output uncertainty predictions on the MNIST test dataset

	KL divergence	*Error of predicted mean*	*Error of predicted variance*	*Computation Time*
Proposed Method	0.093	1.03 %	1.56 %	0.23 sec/image
Layer-wise UT	0.171	1.97 %	3.13 %	0.15 sec/image
Entire-NN UT	1.136	5.38 %	5.79 %	0.38 sec/image

2.4 Summary

In this chapter, a nonlinear uncertainty propagation method based on adaptive Gaussian mixture splitting is developed for artificial neural networks. An innovative nonlinearity evaluation criterion based on the Kullback–Leibler divergence between the true transformation of a componential Gaussian distribution and its approximation via local linearization is derived to select the components for splitting. The splitting will only be applied to those Gaussian components which are carrying a high magnitude of uncertainty (large covariance), have more contribution to the mixture (large weights), and encounter more severe nonlinearity from the activation layer. A set of univariate Gaussian splitting schemes are established, which can be extended to decompose multivariate Gaussian distributions without degradation of accuracy. The nonlinearity examination, Gaussian splitting, and unscented-transform-based propagation of examined Gaussian components are all utilizing the same set of singular value decompositions of covariance matrices, and a Gaussian mixture reduction routine is adopted to further regulate the

number of components and minimize the computation cost.

Three examples of nonlinear dynamic systems approximated by recurrent networks and two data-driven examples modeled by multilayer networks are presented to validate the effectiveness of the proposed method. By comparing the results with Monte Carlo simulations and other methods in the literature, the proposed scheme is proved to be capable of accurately propagating Gaussian mixture uncertainties through neural networks to predict output distributions. The computation times of the proposed method are considerably shorter than the Monte Carlo simulation times and the predicted Gaussian mixtures are more compact than those from the existing Gaussian mixture methods. Therefore, it could be used as a versatile scheme to address both uncertainty propagation through time in recurrent neural networks and layer-wise propagation in deep neural networks.

3. ADAPTIVE GAUSSIAN MIXTURE FILTER FOR NONLINEAR STATE ESTIMATION

This chapter develops an adaptive Gaussian mixture filter (AGMF) with active nonlinearity assessment and Gaussian mixture refinement. For state prediction, a feedforward neural network is used to approximate the state equation and thus the complex dynamics of the system is converted to tractable sigmoid activation functions. Then the nonlinearity in state transition is assessed using the Kullback–Leibler criterion from Chapter 2 at the network's hidden layer so that the Gaussian mixture can be refined before predicting the prior PDF. For Bayesian update, the measurement nonlinearity is assessed by the divergence between true and approximated likelihoods so that the prior can be refined before estimating the posterior. The convergence rates of designed nonlinearity measures and the bound of state PDF estimation errors are quantified. The proposed AGMF is compared with the widely used nonlinear Kalman filters, particle filters, and latest GMFs in the literature on multiple examples, which proves that the proposed filter can achieve state-of-the-art accuracy with a reasonable computational cost on highly nonlinear state estimation problems subject to high magnitudes of uncertainties.

3.1 Preliminaries and Problem Statement

3.1.1 Recursive Bayesian State Estimation

This study considers the state estimation of the following nonlinear dynamic systems in the discrete-time domain:

$$\begin{aligned} \mathbf{x}_k &= \mathbf{f}\left(\mathbf{x}_{k-1}, \mathbf{u}_k, \mathbf{w}_k\right) \\ \mathbf{y}_k &= \mathbf{g}\left(\mathbf{x}_k\right) + \mathbf{v}_k \end{aligned} \tag{3.1}$$

where k=1, 2, … is the discrete-time index, $\mathbf{x}_k$ is the n-dimensional state vector at time k, $\mathbf{u}_k$ is the n_u-dimensional vector of control input, $\mathbf{y}_k$ is the n_y-dimensional vector of measurable output, $\mathbf{w}_k$ is the n_w-dimensional process noise and $\mathbf{v}_k$ is the n_v-dimensional measurement noise. For simplicity of derivation, both $\mathbf{w}_k$ and $\mathbf{v}_k$ are assumed to be zero-mean Gaussian white noises, i.e., $\mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_k)$ and $\mathbf{v}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_k)$ with $\mathbf{Q}_k$ and $\mathbf{R}_k$ be the covariance matrices. The vector-valued nonlinear functions **f** and **g** are the state transition function and measurement function respectively, both of which are assumed to be continuous and **g** is assumed to be differentiable.

In recursive Bayesian estimation, assume that the probability density function (PDF) of state variables based on the measurements up to time $k-1$ is known as $p(\mathbf{x}_{k-1}|\mathbf{y}_{1:k-1})$, then the inference of state PDF at time k consists of two steps. Firstly, a prior PDF is predicted using the state transition integral below to track the evolution of state under system dynamics:

$$p(\mathbf{x}_k \mid \mathbf{y}_{1:k-1}) = \int_{-\infty}^{\infty} p(\mathbf{x}_k \mid \mathbf{x}_{k-1})\, p(\mathbf{x}_{k-1} \mid \mathbf{y}_{1:k-1})\, d\mathbf{x}_{k-1} \tag{3.2}$$

where $p(\mathbf{x}_k|\mathbf{x}_{k-1})$ is the conditional state transition PDF. If the process noise is an additive Gaussian noise, then $p(\mathbf{x}_k|\mathbf{x}_{k-1})=\mathcal{N}(\mathbf{x}_k|\mathbf{f}(\mathbf{x}_{k-1},\mathbf{u}_k), \mathbf{Q}_k)$. However, because the function $\mathbf{f}$ is nonlinear, the state transition PDF is still non-Gaussian with respect to $\mathbf{x}_{k-1}$ and hence the above integral is intractable, even if $p(\mathbf{x}_{k-1}|\mathbf{y}_{1:k-1})$ is Gaussian. Secondly, upon the receiving of a measurement $\mathbf{y}_k$, the Bayes' rule is used to update the prior PDF and calculate the posterior PDF $p(\mathbf{x}_k|\mathbf{y}_{1:k})$:

$$p(\mathbf{x}_k \mid \mathbf{y}_{1:k}) = \frac{p(\mathbf{y}_k \mid \mathbf{x}_k)\, p(\mathbf{x}_k \mid \mathbf{y}_{1:k-1})}{\int_{-\infty}^{\infty} p(\mathbf{y}_k \mid \mathbf{x}_k)\, p(\mathbf{x}_k \mid \mathbf{y}_{1:k-1}) d\mathbf{x}_k} \tag{3.3}$$

where $p(\mathbf{y}_k|\mathbf{x}_k)$ is the conditional likelihood function. This posterior PDF at k can be sent to Eq. (3.2) to predict the prior PDF at the next time step $k+1$, and repeat recursively. However, even with a Gaussian prior PDF, the posterior PDF in Eq. (3.3) may not have a closed-form solution because the likelihood function $p(\mathbf{y}_k|\mathbf{x}_k)=\mathcal{N}(\mathbf{y}_k|\mathbf{g}(\mathbf{x}_k), \mathbf{R}_k)$ is non-Gaussian with respect to $\mathbf{x}_k$.

As can be seen, the intractability of state prediction and Bayesian update is mainly caused by the nonlinearity in the functions $\mathbf{f}$ and $\mathbf{g}$. If $\mathbf{f}$ and $\mathbf{g}$ are only weakly nonlinear so that they can be accurately approximated by their first-order Taylor expansions, then the state transition PDF and measurement likelihood can be written as Gaussian functions with respect to the state $\mathbf{x}_k$. Thereby, both the Eq. (3.2) and (3.3) will have analytical Gaussian-form solutions [73].

3.1.2 Gaussian Mixture Filter

Since the EKF and other high-order Kalman filters (e.g., UKF, QKF) all estimate the state PDF in Gaussian form, their fidelity will depend on the severity of nonlinearity in the system [80]. In contrast, the Gaussian mixture filter (GMF) is designed with the capacity to characterize any non-Gaussian state PDFs. For simplicity of notation, we omit the conditioning on measurements and write the posterior state PDF in a GMF as:

$$p(\mathbf{x}_k) \triangleq p(\mathbf{x}_k \mid \mathbf{y}_{1:k}) = \sum_{i=1}^{N_k} w_k^i p^i(\mathbf{x}_k) \quad \text{with}$$
$$p^i(\mathbf{x}_k) = \mathcal{N}(\mathbf{x}_k \mid \boldsymbol{\mu}_k^i, \boldsymbol{\Sigma}_k^i) = \frac{1}{\sqrt{(2\pi)^n \det(\boldsymbol{\Sigma}_k^i)}} e^{-\frac{1}{2}(\mathbf{x}_k - \boldsymbol{\mu}_k^i)^T \boldsymbol{\Sigma}_k^{i-1}(\mathbf{x}_k - \boldsymbol{\mu}_k^i)} \tag{3.4}$$

where N_k is the number of components at time k, $w_{x,k}^i$ and $\mathcal{N}(\mathbf{x}_k \mid \boldsymbol{\mu}_{x,k}^i, \boldsymbol{\Sigma}_{x,k}^i)$ are the weight and Gaussian density of the i-th component, in which $\boldsymbol{\mu}_{x,k}^i \in \mathbb{R}^n$ and $\boldsymbol{\Sigma}_{x,k}^i \in \mathbb{R}^{n \times n}$ is the mean vector and covariance matrix respectively. When k=0, $p(\mathbf{x}_0)$ represents the initial state uncertainty. Besides, the Gaussian mixture of prior PDF is denoted by the superscript '–' as:

$$p^-(\mathbf{x}_k) \triangleq p(\mathbf{x}_k \mid \mathbf{y}_{1:k-1}) = \sum_{i=1}^{N_k^-} w_k^{i-} p^{i-}(\mathbf{x}_k) = \sum_{i=1}^{N_k^-} w_k^{i-} \mathcal{N}(\mathbf{x}_k \mid \boldsymbol{\mu}_k^{i-}, \boldsymbol{\Sigma}_k^{i-}) \tag{3.5}$$

Given a posterior mixture PDF at k−1 with N_{k-1} components, substituting Eq. (3.5) into Eq. (3.2) will result in a prior PDF with unchanged mixture size and weights:

$$p^-(\mathbf{x}_k) = \sum_{i=1}^{N_{k-1}} w_{k-1}^i \int_{-\infty}^{\infty} p(\mathbf{x}_k \mid \mathbf{x}_{k-1}) p^i(\mathbf{x}_{k-1}) d\mathbf{x}_{k-1} \tag{3.6}$$

For a given prior mixture PDF with N_k^- components, substituting Eq. (3.6) into Eq. (3.3) will also result in a posterior mixture PDF with the same number of components:

$$p(\mathbf{x}_k) = \frac{\sum_{i=1}^{N_k^-} w_k^{i-} p(\mathbf{y}_k \mid \mathbf{x}_k) p^{i-}(\mathbf{x}_k)}{\sum_{i=1}^{N_k^-} w_k^{i-} \int_{-\infty}^{\infty} p(\mathbf{y}_k \mid \mathbf{x}_k) p^{i-}(\mathbf{x}_k) d\mathbf{x}_k} = \sum_{i=1}^{N_k^-} w_k^i p^i(\mathbf{x}_k) \tag{3.7}$$

where the updated posterior density in each component is defined as:

$$p^i(\mathbf{x}_k) = \frac{p(\mathbf{y}_k \mid \mathbf{x}_k) p^{i-}(\mathbf{x}_k)}{\int_{-\infty}^{\infty} p(\mathbf{y}_k \mid \mathbf{x}_k) p^{i-}(\mathbf{x}_k) d\mathbf{x}_k} \tag{3.8}$$

The integral on the denominator is a normalization factor called the marginal likelihood, which reflects the possibility that the measurement $\mathbf{y}_k$ is generated by the state in $p^{i-}(\mathbf{x}_k)$. The posterior weights $w^i{}_k$ are renormalized according to marginal likelihoods so that the components closer to the actual measurement will become more dominant:

$$w_k^i = \frac{w_k^{i-} \int_{-\infty}^{\infty} p(\mathbf{y}_k \mid \mathbf{x}_k) p^{i-}(\mathbf{x}_k) d\mathbf{x}_k}{\sum_{i=1}^{N_k^-} w_k^{i-} \int_{-\infty}^{\infty} p(\mathbf{y}_k \mid \mathbf{x}_k) p^{i-}(\mathbf{x}_k) d\mathbf{x}_k} \tag{3.9}$$

At any time step, the estimated state of a GMF in the sense of minimum mean square error (MMSE) can be easily extracted as the mean and covariance of the Gaussian mixture:

$$\begin{aligned}\hat{\mathbf{x}}_k &= \boldsymbol{\mu}_k = E\left[\mathbf{x}_k\right] = \sum_{i=1}^{N_k} w_k^i \boldsymbol{\mu}_k^i \\ \hat{\boldsymbol{\Sigma}}_k &= E\left[\left(\mathbf{x}_k - \hat{\mathbf{x}}_k\right)\left(\mathbf{x}_k - \hat{\mathbf{x}}_k\right)^T\right] = \sum_{i=1}^{N_k}\left[\boldsymbol{\Sigma}_k^i + \left(\boldsymbol{\mu}_k^i - \hat{\mathbf{x}}_k\right)\left(\boldsymbol{\mu}_k^i - \hat{\mathbf{x}}_k\right)^T\right]\end{aligned} \tag{3.10}$$

The filter performance in this work is thus mainly rated by the mean square error (MSE). Besides, as the GMF is designed to accurately track the state PDF, the Kullback–Leibler (KL) divergence and L^1 norm are used to measure the distance between the true and estimated state PDFs:

$$D_{KL}\left(p \,\|\, \hat{p}\right) = \int_{-\infty}^{\infty} p\left(\mathbf{x}_k\right) \log \frac{p\left(\mathbf{x}_k\right)}{\hat{p}\left(\mathbf{x}_k\right)} d\mathbf{x}_k \tag{3.11}$$

$$\left\| p - \hat{p} \right\| = \int_{-\infty}^{\infty} \left| p\left(\mathbf{x}_k\right) - \hat{p}\left(\mathbf{x}_k\right) \right| d\mathbf{x}_k \tag{3.12}$$

While the KL divergence has been introduced in Eq. (2.6), it is refined here for the PDF of state vector $\mathbf{x}_k$. These two measures both belong to the *f*-divergence family and hence are invariant to invertible transformations [217]. The KL divergence is easy to calculate for exponential family distributions including the Gaussian distribution, and the L^1 norm satisfies the triangle inequality that facilitates the error decomposition, which will be useful in the error analysis.

For a given dynamic system whose model can not be altered, the way to minimize nonlinear distortion in a GMF is to keep the covariance of each component narrow so that the linearization of the system model at a Gaussian center can provide a close approximation within the range of that component. Therefore, how to determine whether a Gaussian component is narrow enough and how to narrow down the wide components without causing misrepresentation in state PDF will be the major challenges to be addressed in the next section.

3.2 The Adaptive Gaussian Mixture Filter

Considering that the nonlinear distortion that deteriorates filter fidelity could occur in both the state prediction and Bayesian update stages, the Gaussian mixtures in both prior and posterior PDFs should be refined so that their components can be assured of high fidelity when processed by Gaussian filters. The nonlinearity assessment and adaptive splitting scheme, which is adopted from Chapter 2, is applied to state prediction in Section 3.2.1. Then this scheme is extended to the Bayesian update in Section 3.2.2. A quantitative analysis of the filter performance is provided in Section 3.2.3.

3.2.1 State Prediction using Gaussian Mixture Uncertainty Propagation

In a nonlinear system, while the measurement function based on sensor principles may fall into certain normally used forms (e.g., the L^2 norm for distance measurement and arctan for angle measurement in radar tracking), the state equation defining system dynamics can be more diverse and complex, which makes the state transition integral intricate. If there is no process noise, the state transition PDF will be a Dirac delta function, i.e., $p(\mathbf{x}_k|\mathbf{x}_{k-1}) = \delta[\mathbf{x}_k - \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{u}_k)]$. However, even in this case, the prior PDF may still not be analytically solvable, especially if the function **f** is not bijective. Hence, though it is known that the prior PDF will be non-Gaussian when **f** is nonlinear, most of the GMFs are still assessing the impact of nonlinearity without explicitly probing the non-Gaussianity. To devise a better nonlinearity assessment scheme, this work considers converting the state transition function **f** into more tractable forms. If function **f** is differentiable and monotonic, then the prior PDF without process noise can be calculated as:

$$\begin{aligned} p^{-}(\mathbf{x}_k) &= \int_{-\infty}^{\infty} \delta\left[\mathbf{x}_k - \mathbf{f}(\mathbf{x}_{k-1})\right] p(\mathbf{x}_{k-1})\, d\mathbf{x}_{k-1} \\ &= \int_{-\infty}^{\infty} \left| \frac{d\mathbf{x}_{k-1}}{d\mathbf{f}(\mathbf{x}_{k-1})} \right| \delta\left[\mathbf{x}_k - \mathbf{f}(\mathbf{x}_{k-1})\right] p(\mathbf{x}_{k-1})\, d\mathbf{f}(\mathbf{x}_{k-1}) \\ &= \left| \frac{d\mathbf{x}_{k-1}}{d\mathbf{f}(\mathbf{x}_{k-1})} \right| p(\mathbf{x}_{k-1}) \Bigg|_{\mathbf{f}(\mathbf{x}_{k-1})=\mathbf{x}_k} = \left| \frac{d\mathbf{f}^{-1}(\mathbf{x}_k)}{d\mathbf{x}_k} \right| p\left(\mathbf{f}^{-1}(\mathbf{x}_k)\right) \end{aligned} \tag{3.13}$$

where | | denotes the determinant of a matrix, and a change of variables is used from the first to the second line. If **f** is a vector of univariate functions, its Jacobian matrix will be diagonal and the determinant will be easy to compute. One instance that satisfies all these conditions is the sigmoid function, which is widely used as the activation function in artificial neural networks.

Inspired by the tractability of the sigmoid function, the filter in this work is devised to solve the state prediction problem by approximating the state transition function **f** with a feedforward neural network (FFNN). An FFNN with a single sigmoid-function hidden layer and sufficiently many hidden nodes can approximate any continuous function to any degree of accuracy [203]. Hence, the FFNN's error in approximating the function **f** can be minimized by choosing a proper network size, so that it will have a trivial effect on the filter's accuracy. Besides, the FFNN can be trained offline with data collected from experiments on the physical system or simulations of the system model, while it facilitates on-line state prediction. As shown in Eq. (3.13), the state transition integral in a sigmoid function layer is more tractable than that for the original system

model, which enables a more efficient assessment of nonlinearity.

The state prediction is fundamentally an uncertainty propagation problem, i.e., propagating the uncertainty quantified by the posterior state PDF at $k-1$ through the system to predict the prior PDF at k. Therefore, the Gaussian mixture scheme for uncertainty propagation in neural networks developed in Chapter 2 is adopted. Define the augmented state vector as $\mathbf{x}_{a,k-1}=[\mathbf{x}_{k-1}; \mathbf{u}_k; \mathbf{w}_k]$, then the single-hidden-layer FFNN to approximate the state equation can be formulated as:

$$\mathbf{x}_k = \mathbf{W}_2\phi\left(\mathbf{W}_1\mathbf{x}_{a,k-1}+\mathbf{b}_1\right)+\mathbf{b}_2+\boldsymbol{\varepsilon}_k \tag{3.14}$$

where $\mathbf{W}_1 \in \mathbb{R}^{m\times(n+n_u+n_w)}$ and $\mathbf{b}_1 \in \mathbb{R}^m$ are the weight and bias in the hidden layer, m is the number of hidden nodes, $\mathbf{W}_2 \in \mathbb{R}^{n\times m}$ and $\mathbf{b}_2 \in \mathbb{R}^n$ are the weight and bias in the output layer, ϕ is the sigmoid function (e.g. logistic or hyperbolic tangent function), and $\boldsymbol{\varepsilon}_k$ is the approximation error of the FFNN. The impact of error $\boldsymbol{\varepsilon}_k$ will be analyzed in Section 3.2.3. Given the posterior PDF at $k-1$, the normally distributed control input $\mathcal{N}(\mathbf{u}_k, \boldsymbol{\Sigma}_{u,k})$ ($\boldsymbol{\Sigma}_{u,k}=0$ if the controller is deterministic) and the process noise $\mathbf{w}_k$, the joint PDF of the augmented state vector is:

$$\begin{aligned} &p\left(\mathbf{x}_{a,k-1}\right)=\sum\nolimits_{i=1}^{N_{k-1}} w_{a,k-1}^i \mathcal{N}\left(\mathbf{x}_{a,k-1}\,|\,\boldsymbol{\mu}_{a,k-1}^i, \boldsymbol{\Sigma}_{a,k-1}^i\right) \quad \text{with} \\ &w_{a,k-1}^i = w_{k-1}^i \quad \boldsymbol{\mu}_{a,k-1}^i=\left[\boldsymbol{\mu}_{k-1}^i; \mathbf{u}_k; \mathbf{0}\right] \quad \boldsymbol{\Sigma}_{a,k-1}^i=\operatorname{diag}\left[\boldsymbol{\Sigma}_{k-1}^i, \boldsymbol{\Sigma}_{u,k}, \mathbf{Q}_k\right] \end{aligned} \tag{3.15}$$

The algorithm to propagate $p(\mathbf{x}_{a,k-1})$ through the FFNN has been elaborated in Section 2.2. But since the notations of variables are a little different between Chapter 2 and this chapter, some key steps in the propagation are briefly presented below for clarity. First, $p(\mathbf{x}_{a,k-1})$ is mapped to the PDF of the state before activation function by a linear transformation $\mathbf{z}=\mathbf{W}_1\mathbf{x}_{a,k-1}+\mathbf{b}_1$:

$$\begin{aligned} &p\left(\mathbf{z}\right)=\sum\nolimits_{i=1}^{N_{k-1}} w_z^i \mathcal{N}\left(\mathbf{z}\,|\,\boldsymbol{\mu}_x^i, \boldsymbol{\Sigma}_z^i\right) \quad \text{with} \\ &w_z^i = w_{a,k-1}^i \quad \boldsymbol{\mu}_z^i = \mathbf{W}_1\boldsymbol{\mu}_{a,k-1}^i+\mathbf{b}_1 \quad \boldsymbol{\Sigma}_z^i = \mathbf{W}_1\boldsymbol{\Sigma}_{a,k-1}^i\mathbf{W}_1^T \end{aligned} \tag{3.16}$$

where $\mathbf{z}$ is the m-dimensional vector of the FFNN's pre-activation states. This linear mapping is equivalent to the state transition integral in Eq. (3.6) with a linear function $\mathbf{f}$.

Then for the Gaussian density in the i-th component of $p(\mathbf{z})$, i.e., $p^i(\mathbf{z}) = \mathcal{N}(\boldsymbol{\mu}_z^i, \boldsymbol{\Sigma}_z^i)$, its true post-activation density $p^i(\mathbf{s})$ and the density approximated by linearization $\hat{p}^i(\mathbf{s})$ will be in the same form as Eq. (2.8), where $\mathbf{s}$ is the m-dimensional vector of the FFNN's post-activation states. The difference is that this Chapter uses m to denote the number of hidden nodes and ϕ to denote the sigmoid function, while Chapter 2 uses k and f respectively. The nonlinearity in ϕ is assessed by the KL divergence between the true and approximated post-activation densities:

$$D_{KL}\left(p^i(\mathbf{s})\,\|\,\hat{p}^i(\mathbf{s})\right) = -\frac{m}{2} + \log\left|\mathbf{A}^i\right| + E_{p^i(\mathbf{z})}\left[\frac{1}{2}\mathcal{F}(\mathbf{z}) - \log\left|d\phi(\mathbf{z})/d\mathbf{z}\right|\right] \quad \text{with}$$
$$\mathcal{F}(\mathbf{z}) = \left(\phi(\mathbf{z}) - \phi\left(\boldsymbol{\mu}_z^i\right)\right)^T \left(\mathbf{A}^i\boldsymbol{\Sigma}_z^i\mathbf{A}^{iT}\right)^{-1}\left(\phi(\mathbf{z}) - \phi\left(\boldsymbol{\mu}_z^i\right)\right) \tag{3.17}$$

The Eq. (3.17) is a reorganization of Eq. (2.9) and the expectation integral can be evaluated using UT. For the *m*-dimensional Gaussian density $p^i(\mathbf{z})$, $2m+1$ sigma points Z_j^i and their weights W_j^i (j=1,…,$2m$+1) can be generated using Eq. (2.10) based on the SVD in Eq. (2.11). Then the expectation integral in Eq. (3.17) can be evaluated as:

$$E_{p^i(\mathbf{z})}\left[0.5\mathcal{F}(\mathbf{z}) - \log\left|\phi'(\mathbf{z})\right|\right] = \sum_{j=0}^{2m} W_j^i\left[0.5\mathcal{F}\left(Z_j^i\right) - \log\left|\phi'\left(Z_j^i\right)\right|\right] \tag{3.18}$$

If the KL divergence exceeds a threshold th_D, it means the pre-activation density $p^i(\mathbf{z})$ will be noticeably distorted and its post-activation density can not be precisely approximated in Gaussian form. Hence, this component will be split into a set of narrower sub-components to alleviate the distortion. For a weighted component $w_z^i p^i(\mathbf{z})$ with density $p^i(\mathbf{z})=\mathcal{N}(\boldsymbol{\mu}_z^i, \boldsymbol{\Sigma}_z^i)$, the splitting along its j-th singular vector can be obtained as a sub-mixture of P components:

$$\sum_{p=1}^{P} \tilde{w}_z^{i,p}\mathcal{N}\left(\mathbf{z}\,|\,\tilde{\boldsymbol{\mu}}_z^{i,p}, \tilde{\boldsymbol{\Sigma}}_z^{i,p}\right) \cong w_z^i\mathcal{N}\left(\mathbf{z}\,|\,\boldsymbol{\mu}_z^i, \boldsymbol{\Sigma}_z^i\right) \quad \text{with} \quad \tilde{w}_z^{i,p} = w_z^i\tilde{w}_p$$
$$\tilde{\boldsymbol{\mu}}_z^{i,p} = \boldsymbol{\mu}_z^i + \tilde{\mu}_p\sqrt{\lambda_j}\,\mathbf{v}_j \quad \tilde{\boldsymbol{\Sigma}}_z^{i,p} = \mathbf{V}\tilde{\boldsymbol{\Lambda}}\mathbf{V} \quad \tilde{\boldsymbol{\Lambda}} = \text{diag}\left[\lambda_1 \quad \ldots \quad \tilde{\sigma}^2\lambda_j \quad \ldots \quad \lambda_m\right] \tag{3.19}$$

where $\tilde{w}_p$, $\tilde{\mu}_p$, $\tilde{\sigma}$ are the weights, means, and shared standard deviation of sub-components in a splitting scheme from Table 2-2. In this work, the splitting is performed along the singular vector on which the sigma points have the largest norm of deviation from a linear extrapolation:

$$\max_j \frac{\left\|\phi\left(Z_j^i\right) - \phi\left(\boldsymbol{\mu}_z^i\right)\right\|}{\left\|\mathbf{A}^i\left(Z_j^i - \boldsymbol{\mu}_z^i\right)\right\|} + \frac{\left\|\phi\left(Z_{j+m}^i\right) - \phi\left(\boldsymbol{\mu}_z^i\right)\right\|}{\left\|\mathbf{A}^i\left(Z_{j+m}^i - \boldsymbol{\mu}_z^i\right)\right\|} \quad j \in \{1,\ldots,m\} \tag{3.20}$$

If the pre-activation PDF is a Gaussian mixture, this nonlinearity assessment and splitting process will be repeated on each component until the KL divergences of all the components drop below the threshold. Then the refined mixture $\tilde{p}(\mathbf{z}) = \sum_{i=1}^{N_k^-}\tilde{w}_z^i\mathcal{N}\left(\mathbf{z}\,|\,\tilde{\boldsymbol{\mu}}_z^i, \tilde{\boldsymbol{\Sigma}}_z^i\right)$, of which the number of components has increased from N_{k-1} to N_k^-, can be propagated using UT as:

$$\hat{p}(\mathbf{s}) = \sum_{i=1}^{N_k^-} w_s^i \mathcal{N}\left(\mathbf{s} \mid \boldsymbol{\mu}_s^i, \boldsymbol{\Sigma}_s^i\right) \quad w_s^i = \tilde{w}_z^i$$
$$\boldsymbol{\mu}_s^i = \sum_{j=0}^{2m} W_j^i \boldsymbol{\phi}\left(Z_j^i\right) \tag{3.21}$$
$$\boldsymbol{\Sigma}_s^i = \sum_{j=0}^{2m} W_j^i \left(\boldsymbol{\phi}\left(Z_j^i\right) - \boldsymbol{\mu}_s^i\right)\left(\boldsymbol{\phi}\left(Z_j^i\right) - \boldsymbol{\mu}_s^i\right)^T$$

Next, the linear transformation $\mathbf{x}_k = \mathbf{W}_2\mathbf{s} + \mathbf{b}_2$ is applied to this post-activation PDF, and the prior state PDF $p^-(\mathbf{x}_k)$ can be predicted with the following Gaussian mixture parameters:

$$w_k^{i-} = w_s^i \quad \boldsymbol{\mu}_k^{i-} = \mathbf{W}_2\boldsymbol{\mu}_s^i + \mathbf{b}_2 \quad \boldsymbol{\Sigma}_k^{i-} = \mathbf{W}_2\boldsymbol{\Sigma}_s^i\mathbf{W}_2^T \tag{3.22}$$

Lastly, a Gaussian mixture reduction step, such as the one described in Section 2.2.4, can be applied to merge components and prevent the splitting operation from increasing the mixture size exponentially over time, which is necessary to maintain the computational efficiency.

3.2.2 Bayesian Measurement Update with Adaptive Refinement

A Bayesian state estimator incorporates measurements from the actual system to correct the prior PDFs predicted based on the system model. Applying the uncertainty propagation method in Section 3.2.1 to the measurement equation can provide a refined prediction of measurement PDF $p(\mathbf{y}_k)$, but may not explicitly improve the fidelity of the posterior state PDF. Therefore, a nonlinearity assessment and Gaussian mixture refinement scheme dedicated to the Bayesian update should be designed. As shown in Eq. (3.8), the shape of each posterior density is defined by the product of its corresponding prior density and the measurement likelihood. Therefore, given a Gaussian mixture prior PDF, the impact of nonlinearity is mainly associated with the likelihood function [76].

With the additive Gaussian measurement noise, the true likelihood function can be written as $p(\mathbf{y}_k|\mathbf{x}_k) = \mathcal{N}(\mathbf{y}_k|\mathbf{g}(\mathbf{x}_k), \mathbf{R}_k)$, which is non-Gaussian with respect to $\mathbf{x}_k$ as the function $\mathbf{g}$ is nonlinear. If $\mathbf{g}$ is differentiable, the approximated likelihood by linearizing $\mathbf{g}$ at the i-th prior Gaussian density $p^{i-}(\mathbf{x}_k) = \mathcal{N}(\boldsymbol{\mu}_k^{i-}, \boldsymbol{\Sigma}_k^{i-})$ can be written as a Gaussian function of $\mathbf{x}_k$:

$$\hat{p}\left(\mathbf{y}_k \mid \mathbf{x}_k\right) = \mathcal{N}\left(\mathbf{y}_k \mid \mathbf{g}\left(\boldsymbol{\mu}_k^{i-}\right) + \mathbf{H}_k^i\left(\mathbf{x}_k - \boldsymbol{\mu}_k^{i-}\right), \mathbf{R}_k\right)$$
$$\propto \mathcal{N}\left(\mathbf{x}_k \mid \boldsymbol{\mu}_k^{i-} + \mathbf{H}_k^{i\,-1}\left(\mathbf{y}_k - \mathbf{g}\left(\boldsymbol{\mu}_k^{i-}\right)\right), \mathbf{H}_k^{i\,-1}\mathbf{R}_k\mathbf{H}_k^{i\,-T}\right) \tag{3.23}$$
$$\mathbf{H}_k^i = \left[\frac{d\mathbf{g}(\mathbf{x}_k)}{d\mathbf{x}_k}\right]_{\mathbf{x}_k = \boldsymbol{\mu}_k^{i-}}$$

where the matrix $\mathbf{H}_k^i$ should be invertible or have a pseudoinverse. This likelihood makes the Bayesian update analytically solvable (i.e., the EKF solution). How well this likelihood can approximate the true likelihood depends on the severity of nonlinearity of **g** within the span of $p^{i-}(\mathbf{x}_k)$. Hence, the following divergence between the true and approximated likelihoods is defined to assess the impact of nonlinearity in the Bayesian measurement update:

$$\begin{aligned}
&\int_{-\infty}^{\infty} p^{i-}\left(\mathbf{x}_k\right)\left|\log\frac{p\left(\mathbf{y}_k\mid\mathbf{x}_k\right)}{\hat{p}\left(\mathbf{y}_k\mid\mathbf{x}_k\right)}\right|d\mathbf{x}_k=\frac{1}{2}E_{p^{i-}(\mathbf{x}_k)}\left[\left|\mathcal{G}\left(\mathbf{x}_k\right)-\hat{\mathcal{G}}\left(\mathbf{x}_k\right)\right|\right]\\
&\text{with}\\
&\mathcal{G}\left(\mathbf{x}_k\right)=\left(\mathbf{g}\left(\mathbf{x}_k\right)-\mathbf{y}_k\right)^T\mathbf{R}_k^{-1}\left(\mathbf{g}\left(\mathbf{x}_k\right)-\mathbf{y}_k\right)\\
&\hat{\mathcal{G}}\left(\mathbf{x}_k\right)=\left(\mathbf{H}_k^i\mathbf{x}_k-\mathbf{H}_k^i\boldsymbol{\mu}_k^{i-}+\mathbf{g}\left(\boldsymbol{\mu}_k^{i-}\right)-\mathbf{y}_k\right)^T\mathbf{R}_k^{-1}\left(\mathbf{H}_k^i\mathbf{x}_k-\mathbf{H}_k^i\boldsymbol{\mu}_k^{i-}+\mathbf{g}\left(\boldsymbol{\mu}_k^{i-}\right)-\mathbf{y}_k\right)
\end{aligned}\tag{3.24}$$

For a Gaussian mixture prior PDF, the above likelihood divergence can be estimated for each of its *n*-dimensional Gaussian density $p^{i-}(\mathbf{x}_k)$ using UT as:

$$E_{p^{i-}(\mathbf{x}_k)}\left[\left|\mathcal{G}\left(\mathbf{x}_k\right)-\hat{\mathcal{G}}\left(\mathbf{x}_k\right)\right|\right]=\sum\nolimits_{j=0}^{2n}W_j^i\left|\mathcal{G}\left(X_j^i\right)-\hat{\mathcal{G}}\left(X_j^i\right)\right|\tag{3.25}$$

where X_j^i and W_j^i are the sigma points and weights (j=1,…,2n+1) of the i-th component generated using Eq. (2.10) based on the SVD of $\boldsymbol{\Sigma}_k^{i-}$ as in Eq. (2.11). If the likelihood divergence of a prior component exceeds a threshold, this component will be split into narrower sub-components in the same way as Eq. (3.19), along the singular vector with the largest directional divergence:

$$\max_j\left|\mathcal{G}\left(X_j^i\right)-\hat{\mathcal{G}}\left(X_j^i\right)\right|+\left|\mathcal{G}\left(X_{j+n}^i\right)-\hat{\mathcal{G}}\left(X_{j+n}^i\right)\right|\quad j\in\{1,\ldots,n\}\tag{3.26}$$

By repeating the splitting process until the likelihood divergences of all the components drop below a threshold th_L, the refined prior PDF is obtained $\tilde{p}^-\left(\mathbf{x}_k\right)=\sum_{i=1}^{N_k}\tilde{w}_k^{i-}\mathcal{N}\left(\mathbf{x}_k\mid\tilde{\boldsymbol{\mu}}_k^{i-},\tilde{\boldsymbol{\Sigma}}_k^{i-}\right)$, in which the number of components has increased from N_k^- to N_k. The refined prior densities should have been amply narrow to yield near-Gaussian posterior densities. Given that the linearization of **g** and sigma points are both available, either EKF or UKF can be applied to each component. The UKF is adopted in this work as it is in general more accurate than EKF. The posterior PDF can be estimated using UKF with the following Gaussian mixture parameters:

$$\begin{aligned}
w_k^i&=\frac{\tilde{w}_k^{i-}\mathcal{N}\left(\mathbf{y}_k\mid\bar{\mathbf{y}}_k^i,\boldsymbol{\Sigma}_{k,yy}^i\right)}{\sum_{i=1}^{N_k^-}\tilde{w}_k^{i-}\mathcal{N}\left(\mathbf{y}_k\mid\bar{\mathbf{y}}_k^i,\boldsymbol{\Sigma}_{k,yy}^i\right)}\\
\boldsymbol{\mu}_k^i&=\tilde{\boldsymbol{\mu}}_k^{i-}+\mathbf{K}_k^i\left(\mathbf{y}_k-\bar{\mathbf{y}}_k^i\right)\\
\boldsymbol{\Sigma}_k^i&=\tilde{\boldsymbol{\Sigma}}_k^{i-}-\mathbf{K}_k^i\boldsymbol{\Sigma}_{k,yy}^i\mathbf{K}_k^{iT}
\end{aligned}\tag{3.27}$$

where the weights are renormalized using Eq. (3.9), and the UKF gains $\mathbf{K}_k^i$ are obtained as [66]:

$$\begin{aligned}
&\bar{\mathbf{x}}_k^i = \sum_{j=0}^{2n} W_j^i X_j^i \quad \bar{\mathbf{y}}_k^i = \sum_{j=0}^{2n} W_j^i \mathbf{g}\left(X_j^i\right) \\
&\boldsymbol{\Sigma}_{k,xy}^i = \sum_{j=0}^{2n} W_j^i \left[X_j^i - \bar{\mathbf{x}}_k^i\right]\left[\mathbf{g}\left(X_j^i\right) - \bar{\mathbf{y}}_k^i\right]^T \\
&\boldsymbol{\Sigma}_{k,yy}^i = \sum_{j=0}^{2n} W_j^i \left[\mathbf{g}\left(X_j^i\right) - \bar{\mathbf{y}}_k^i\right]\left[\mathbf{g}\left(X_j^i\right) - \bar{\mathbf{y}}_k^i\right]^T \\
&\mathbf{K}_k^i = \boldsymbol{\Sigma}_{k,xy}^i \boldsymbol{\Sigma}_{k,yy}^{i\ -1}
\end{aligned} \tag{3.28}$$

Lastly, the Gaussian mixture reduction method in Section 2.2.4 can be applied again to simplify the posterior PDF if the mixture size exceeds a limit N_{max}, before proceeding to the next time step k+1. The complete AGMF algorithm, including both state prediction and Bayesian update, is summarized in Algorithm II.

Algorithm II: adaptive Gaussian mixture filter (AGMF)

Given the system model and the initial uncertainty $p\left(\mathbf{x}_0\right) = \sum_{i=1}^{N_0} w_0^i \mathcal{N}\left(\mathbf{x}_0 \mid \boldsymbol{\mu}_0^i, \boldsymbol{\Sigma}_0^i\right)$, set k=1.

(1) Training of the FFNN in Eq. (3.14) by backpropagation to approximate the function **f**;

State Prediction for the Prior PDF

(2) Linear transformation from $p(\mathbf{x}_{a,k-1})$ to $p(\mathbf{z})$ using Eq. (3.16);

(3) For each component in $p(\mathbf{z})$, evaluate Eq. (3.17) using UT;

(4) If $D_{KL}\left(p^i\left(\mathbf{s}\right) \| \hat{p}^i\left(\mathbf{s}\right)\right) > th_D$, split the component i using Eq. (3.19);

(5) Repeat steps (3) and (4) until all components drops below th_D;

(6) Linear transformation to predict the prior state PDF $p^-(\mathbf{x}_k)$ using Eq. (3.22);

(7) Gaussian mixture reduction on $p^-(\mathbf{x}_k)$ if the mixture size N_k^- is larger than N_{max};

Bayesian Measurement Update for the Posterior PDF

(8) For each component in $p^-(\mathbf{x}_k)$, evaluate Eq. (3.24) using UT;

(9) If $E_{p^{i-}(\mathbf{x}_k)}\left|\log\left[p\left(\mathbf{y}_k \mid \mathbf{x}_k\right) / \hat{p}\left(\mathbf{y}_k \mid \mathbf{x}_k\right)\right]\right| > th_L$, split the component i;

(10) Repeat steps (8) and (9) until all components drops below th_L;

(11) Apply the UKF in Eq. (3.27) to obtain the posterior state PDF $p(\mathbf{x}_k)$;

(12) Gaussian mixture reduction on $p(\mathbf{x}_k)$ if the mixture size N_k is larger than N_{max};

(13) k=k+1, go to step (2).

3.2.3 The Analysis of the Adaptive Gaussian Mixture Filter

In this section, the performance of the AGMF in estimating the state PDF is analyzed. First, it is shown that the splitting process at the sigmoid hidden layer of an FFNN could refine a

Gaussian density to the level of any threshold value th_D.

Lemma 1. Let the pre-activation Gaussian density $\mathcal{N}(\mathbf{z}|\boldsymbol{\mu}_z, \boldsymbol{\Sigma}_z)$ have a bounded mean vector, i.e., $\|\boldsymbol{\mu}_z\|_\infty < M_\mu < \infty$, and the singular values λ_i (i=1,…,m) of covariance matrix $\boldsymbol{\Sigma}_z$ satisfy $\lambda_{\max}/\lambda_{\min} < M_\lambda < \infty$, then for any arbitrarily small positive value of th_D, there exist $\sigma_D > 0$ such that if the trace $\sqrt{\mathrm{tr}(\boldsymbol{\Sigma}_z)} \le \sigma_D$, the KL divergence in Eq. (3.17) satisfies $D_{KL}\left(p(\mathbf{s}) \| \hat{p}(\mathbf{s})\right) < th_D$.

The proof of Lemma 1 is provided in Appendix C. Since each splitting on a singular value λ_j reduces tr($\boldsymbol{\Sigma}_z$) by $\left(1-\tilde{\sigma}^2\right)\lambda_j$, repeating the splitting will eventually make tr($\boldsymbol{\Sigma}_z$) amply small such that the refined component can yield a KL divergence lower than th_D. Next, the error of the prior PDF predicted based on this splitting method is analyzed.

Lemma 2. Assume the L^1 norm between the true and estimated posterior PDFs at $k-1$ is known $\left\| p(\mathbf{x}_{k-1}) - \hat{p}(\mathbf{x}_{k-1}) \right\| = \delta_{k-1}$ and the FFNN's approximation error is a zero-mean Gaussian process $\boldsymbol{\varepsilon}_k \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_\varepsilon)$, then the L^1 norm error of the predicted prior PDF at k is bounded by:

$$\begin{aligned} &\delta_k^- = \left\| p(\mathbf{x}_k) - \hat{p}^-(\mathbf{x}_k) \right\| \le \delta_{k-1} + \frac{N_k^- - N_{k-1}}{P-1}\sqrt{2\tilde{D}_{KL}} + \sqrt{2th_D} + \delta_{\varepsilon,k} \\ &\text{with} \quad \delta_{\varepsilon,k} = \sum_{i=1}^{N_k^-} w_k^{i-} \sqrt{\mathrm{tr}\left(\left(\boldsymbol{\Sigma}_k^{i-}\right)^{-1}\boldsymbol{\Sigma}_\varepsilon\right) - \log\left|\mathbf{I} + \left(\boldsymbol{\Sigma}_k^{i-}\right)^{-1}\boldsymbol{\Sigma}_\varepsilon\right|} \end{aligned} \tag{3.29}$$

where $\tilde{D}_{KL}$ and P are the KL divergence and number of components in the splitting scheme from Table 2-2, and $N_k^- - N_{k-1}$ is the increase of mixture size due to the splitting to meet th_D.

Proof: Given that the L^1 norm is invariant under linear transformations, the error after Eq. (3.16) is still $\left\| p(\mathbf{z}) - \hat{p}(\mathbf{z}) \right\| = \delta_{k-1}$. Then assume that the components $w_z^i p^i(\mathbf{z})$ in the estimated pre-activation PDF $\hat{p}(\mathbf{z})$ are split into sub-mixtures $w_z^i \sum_{p=1}^{P} \tilde{w}_p \tilde{p}_p^i(\mathbf{z})$ in the refined PDF $\tilde{p}(\mathbf{z})$. By using the triangle inequality of L^1 norm and the Pinsker's inequality that the L^1 norm is bounded by the KL divergence $\|p - \tilde{p}\| \le \sqrt{2D_{KL}(p \| \tilde{p})}$, it can be shown that:

$$\begin{aligned} &\left\| \hat{p}(\mathbf{z}) - \tilde{p}(\mathbf{z}) \right\| = \left\| \sum\nolimits_i w_z^i \left[p^i(\mathbf{z}) - \sum \tilde{w}_p \tilde{p}_p^i(\mathbf{z}) \right] \right\| \\ &\le \sum\nolimits_i w_z^i \left\| p^i(\mathbf{z}) - \sum \tilde{w}_p \tilde{p}_p^i(\mathbf{z}) \right\| \\ &\le \sum\nolimits_i w_z^i \sqrt{2D_{KL}\left(p^i(\mathbf{z}) \| \sum \tilde{w}_p \tilde{p}_p^i(\mathbf{z})\right)} = \sum\nolimits_i w_z^i \sqrt{2\tilde{D}_{KL}} \end{aligned} \tag{3.30}$$

Since each splitting adds $P-1$ new components and the component weight has $w_z^i<1$, if the mixture size is increased by $N_k^- - N_{k-1}$, the total L^1 norm error caused by splitting is bounded by:

$$\left\|\hat{p}(\mathbf{z})-\tilde{p}(\mathbf{z})\right\| \le \sum_i w_z^i \sqrt{2\tilde{D}_{KL}} \le \frac{N_k^- - N_{k-1}}{P-1}\sqrt{2\tilde{D}_{KL}} \tag{3.31}$$

Then to propagation the Gaussian mixture through the sigmoid function:

$$\begin{aligned}&\left\|p(\mathbf{s})-\hat{p}(\mathbf{s})\right\| = \left\|\Phi\left(p(\mathbf{z})\right)-\Phi_A\left(\tilde{p}(\mathbf{z})\right)\right\| \\ &\le \left\|\Phi\left(p(\mathbf{z})\right)-\Phi\left(\tilde{p}(\mathbf{z})\right)\right\| + \left\|\Phi\left(\tilde{p}(\mathbf{z})\right)-\Phi_A\left(\tilde{p}(\mathbf{z})\right)\right\|\end{aligned} \tag{3.32}$$

where Φ is the exact PDF mapping through function ϕ, and Φ_A is the approximated mapping via linearization. As the sigmoid function ϕ is continuous and monotonic, it is an invertible transformation to which the L^1 norm is invariant [217], and thus the L^1 norm between exact mapping solutions is:

$$\begin{aligned}&\left\|\Phi\left(p(\mathbf{z})\right)-\Phi\left(\tilde{p}(\mathbf{z})\right)\right\| = \left\|p(\mathbf{z})-\tilde{p}(\mathbf{z})\right\| \\ &\le \left\|p(\mathbf{z})-\hat{p}(\mathbf{z})\right\| + \left\|\hat{p}(\mathbf{z})-\tilde{p}(\mathbf{z})\right\| = \delta_{k-1} + \frac{N_k^- - N_{k-1}}{P-1}\sqrt{2\tilde{D}_{KL}}\end{aligned} \tag{3.33}$$

Then provided that all the refined components have been narrow enough to yield KL divergences lower than th_D under the approximated mapping, it can be shown that:

$$\begin{aligned}&\left\|\Phi\left(\tilde{p}(\mathbf{z})\right)-\Phi_A\left(\tilde{p}(\mathbf{z})\right)\right\| \le \sum_{i=1}^{N_k^-} \tilde{w}_z^i \left\|\Phi\left(\tilde{p}^i(\mathbf{z})\right)-\Phi_A\left(\tilde{p}^i(\mathbf{z})\right)\right\| \\ &\le \sum_{i=1}^{N_k^-} \tilde{w}_z^i \sqrt{2D_{KL}\left(\Phi\left(\tilde{p}^i\right)\|\,\Phi_A\left(\tilde{p}^i\right)\right)} \le \sum_{i=1}^{N_k^-} \tilde{w}_z^i \sqrt{2th_D} = \sqrt{2th_D}\end{aligned} \tag{3.34}$$

Next, the linear transformation in Eq. (3.22) will not alter the L^1 norm in Eq. (3.32). Lastly, due to the approximation error $\boldsymbol{\varepsilon}_k$, when the state predicted by FFNN is $\mathbf{x}_k \sim \sum w_k^{i-}\mathcal{N}\left(\boldsymbol{\mu}_k^{i-}, \boldsymbol{\Sigma}_k^{i-}\right)$, the state from the actual system should be $\mathbf{x}_k+\boldsymbol{\varepsilon}_k \sim \sum w_k^{i-}\mathcal{N}\left(\boldsymbol{\mu}_k^{i-}, \boldsymbol{\Sigma}_k^{i-}+\boldsymbol{\Sigma}_\varepsilon\right)$. Using the formula for the KL divergence between two Gaussians [218], the impact of error $\boldsymbol{\varepsilon}_k$ can be quantified as:

$$\begin{aligned}\delta_{\varepsilon,k} &= \left\|\hat{p}^-\left(\mathbf{x}_k\right)-\hat{p}^-\left(\mathbf{x}_k+\boldsymbol{\varepsilon}_k\right)\right\| \\ &\le \sum_{i=1}^{N_k^-} w_k^{i-}\sqrt{2D_{KL}\left(\mathcal{N}\left(\boldsymbol{\mu}_k^{i-}, \boldsymbol{\Sigma}_k^{i-}+\boldsymbol{\Sigma}_e\right)\|\,\mathcal{N}\left(\boldsymbol{\mu}_k^{i-}, \boldsymbol{\Sigma}_k^{i-}\right)\right)} \\ &= \sum_{i=1}^{N_k^-} w_k^{i-}\sqrt{\mathrm{tr}\left(\left(\boldsymbol{\Sigma}_k^{i-}\right)^{-1}\boldsymbol{\Sigma}_\varepsilon\right)-\log\left|\mathbf{I}+\left(\boldsymbol{\Sigma}_k^{i-}\right)^{-1}\boldsymbol{\Sigma}_\varepsilon\right|}\end{aligned} \tag{3.35}$$

Combine Eq. (3.33)~(3.35), the L^1 norm error in Lemma 2 can be established. Q.E.D.

The L^1 norm in Eq. (3.29) consists of the error from the previous time step, the error due to

splitting, the error of approximated propagation, and the error introduced by approximating the system with an FFNN. In practice, the error covariance $\mathbf{\Sigma}_\varepsilon$ can be estimated by the MSE in FFNN training, which can be made desirably small given a sufficient number of hidden nodes. Then the magnitude of δ_ε can be minimized. If the splitting scheme is also precise and the Gaussian mixture is refined to a small value of th_D, the total L^1 norm error in the prior PDF can be regulated to a minimal level. Next, the error of the posterior state PDF will be analyzed.

Lemma 3. Assume that 1) the Gaussian prior PDF $p^-(\mathbf{x}_k)=\mathcal{N}(\boldsymbol{\mu}_k^-, \mathbf{\Sigma}_k^-)$ has a finite distance to the measurement $\left\|\mathbf{y}_k-\mathbf{g}\left(\boldsymbol{\mu}_k^-\right)\right\|_2 \le M_g<\infty$; 2) the measurement function $\mathbf{g}$ is differentiable and the norm of its Jacobian matrix is bounded $\|\mathbf{H}_k\|_2 \le M_H<\infty$; 3) Taylor's expansion of $\mathbf{g}$ has a bounded second-order remainder:

$$\mathbf{g}\left(\mathbf{x}_k\right)=\mathbf{g}\left(\boldsymbol{\mu}_k^-\right)+\mathbf{H}_k\left(\mathbf{x}_k-\boldsymbol{\mu}_k^-\right)+\mathbf{r}_2\left(\mathbf{x}_k\right) \quad \text{with} \quad \left\|\mathbf{r}_2\right\|_2 \le M_r\left\|\mathbf{x}_k-\boldsymbol{\mu}_k^-\right\|_2^2 \tag{3.36}$$

then for any arbitrarily small positive value of th_L, there exist $\sigma_L>0$ such that if the trace $\mathrm{tr}(\mathbf{\Sigma}_k^-)<\sigma_L$, the likelihood divergence in Eq. (3.24) will be lower than th_L, and the KL divergence between the exact and estimated posterior PDFs will be lower than $M\times th_L$ for a finite factor $M<\infty$.

The proof of Lemma 3 is provided in Appendix D. A sufficient condition to satisfy Eq. (3.36) is that the function $\mathbf{g}$ is second-order differentiable and its Hessian matrices have bounded 2-norms. Next, the error analysis of posterior PDF is extended to the Gaussian mixture case.

Lemma 4. Assume that 1) the L^1 norm between the true and estimated prior PDFs at k is known as $\left\|p^-\left(\mathbf{x}_k\right)-\hat{p}^-\left(\mathbf{x}_k\right)\right\|=\delta_k^-$; 2) given the L^1 error of initial condition $\left\|p\left(\mathbf{x}_0\right)-\hat{p}\left(\mathbf{x}_0\right)\right\|=\delta_0$ and a sequence of measurements $\{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_k\}$, there exists $C<\infty$ and $\rho<1$ such that the exact solution of filtered state PDF satisfies:

$$\left\|\Gamma_k\left(\Gamma_{k-1}\left(\ldots\Gamma_1\left(p\left(\mathbf{x}_0\right)\right)\ldots\right)\right)-\Gamma_k\left(\Gamma_{k-1}\left(\ldots\Gamma_1\left(\hat{p}\left(\mathbf{x}_0\right)\right)\ldots\right)\right)\right\| \le C\rho^k\delta_0 \tag{3.37}$$

with Γ_k be the PDF mapping under Bayesian update with measurement $\mathbf{y}_k$, then the L^1 norm error of the estimated posterior PDF at k is bounded by:

$$\delta_k=\left\|p\left(\mathbf{x}_k\right)-\hat{p}\left(\mathbf{x}_k\right)\right\| \le C\rho\left(\delta_k^-+\frac{N_k-N_k^-}{P-1}\sqrt{2\tilde{D}_{KL}}\right)+\sqrt{2M\times th_L} \tag{3.38}$$

where M is the finite factor as in Lemma 3 and $N_k - N_k^-$ is the increase of mixture size from prior to posterior PDF due to the splitting to meet th_L.

Proof: First, the predicted prior PDF $\hat{p}^-(\mathbf{x}_k)$ is refined by splitting to get $\tilde{p}^-(\mathbf{x}_k)$. Same as Eq. (3.31), the L^1 norm error after splitting is

$$\begin{aligned}&\left\|p^-(\mathbf{x}_k)-\tilde{p}^-(\mathbf{x}_k)\right\| \le \left\|p^-(\mathbf{x}_k)-\hat{p}^-(\mathbf{x}_k)\right\| + \left\|\hat{p}^-(\mathbf{x}_k)-\tilde{p}^-(\mathbf{x}_k)\right\| \\ &= \delta_k^- + \frac{N_k - N_k^-}{P-1}\sqrt{2\tilde{D}_{KL}}\end{aligned} \tag{3.39}$$

Then to update the prior PDF and get the posterior PDF:

$$\begin{aligned}&\left\|p(\mathbf{x}_k)-\hat{p}(\mathbf{x}_k)\right\| = \left\|\Gamma_{\mathbf{y}_k}\left(p^-(\mathbf{x}_k)\right)-\Gamma_A\left(\tilde{p}^-(\mathbf{x}_k)\right)\right\| \\ &\le \left\|\Gamma\left(p^-(\mathbf{x}_k)\right)-\Gamma\left(\tilde{p}^-(\mathbf{x}_k)\right)\right\| + \left\|\Gamma\left(\tilde{p}^-(\mathbf{x}_k)\right)-\Gamma_A\left(\tilde{p}^-(\mathbf{x}_k)\right)\right\|\end{aligned} \tag{3.40}$$

where Γ is the exact Bayesian update, and Γ_A is the approximated update using the likelihood in Eq. (3.23). Same as Eq. (3.32), while the error is measured between the exact mapping of true PDF and the approximated mapping of estimated PDF, the exact mapping of estimated PDF is used as a medium for error decoupling. Applying Eq. (3.37) to the first decoupled term implies:

$$\left\|\Gamma\left(p^-(\mathbf{x}_k)\right)-\Gamma\left(\tilde{p}^-(\mathbf{x}_k)\right)\right\| \le C\rho\left\|p^-(\mathbf{x}_k)-\tilde{p}^-(\mathbf{x}_k)\right\| \tag{3.41}$$

And the KL divergence between $\Gamma_A\left(\tilde{p}^-(\mathbf{x}_k)\right) = \sum_{i=1}^{N_k}\hat{w}_k^i\hat{p}^i(\mathbf{x}_k)$ and $\Gamma\left(\tilde{p}^-(\mathbf{x}_k)\right) = \sum_{i=1}^{N_k} w_k^i p^i(\mathbf{x}_k)$ is bounded by the weighted sum of componential divergences [218]:

$$D_{KL}\left(\Gamma_A\left(\tilde{p}^-(\mathbf{x}_k)\right)\|\,\Gamma\left(\tilde{p}^-(\mathbf{x}_k)\right)\right) \le \sum_{i=1}^{N_k}\hat{w}_k^i\int_{-\infty}^{\infty}\hat{p}^i(\mathbf{x}_k)\log\frac{\hat{w}_k^i\hat{p}^i(\mathbf{x}_k)}{w_k^i\hat{p}^i(\mathbf{x}_k)}d\mathbf{x}_k \tag{3.42}$$

Substituting the weights and densities in Eq. (3.8) and (3.9) into each component implies:

$$\begin{aligned}&\int_{-\infty}^{\infty}\hat{p}^i(\mathbf{x}_k)\log\frac{\hat{w}_k^i\hat{p}^i(\mathbf{x}_k)}{w_k^i p^i(\mathbf{x}_k)}d\mathbf{x}_k \\ &= \int_{-\infty}^{\infty}\hat{p}^i(\mathbf{x}_k)\log\frac{\hat{p}(\mathbf{y}_k\,|\,\mathbf{x}_k)}{p(\mathbf{y}_k\,|\,\mathbf{x}_k)}d\mathbf{x}_k + \log\frac{\sum_{j=1}^{N_k}\tilde{w}_k^{j-}\int_{-\infty}^{\infty}p(\mathbf{y}_k\,|\,\mathbf{x}_k)\,\tilde{p}^{j-}(\mathbf{x}_k)d\mathbf{x}_k}{\sum_{j=1}^{N_k}\tilde{w}_k^{j-}\int_{-\infty}^{\infty}\hat{p}(\mathbf{y}_k\,|\,\mathbf{x}_k)\,\tilde{p}^{j-}(\mathbf{x}_k)d\mathbf{x}_k} \\ &\le E_{\hat{p}^i(\mathbf{x}_k)}\left[\left|\frac{\hat{p}(\mathbf{y}_k\,|\,\mathbf{x}_k)}{p(\mathbf{y}_k\,|\,\mathbf{x}_k)}\right|\right] + \max_j\log\frac{\int_{-\infty}^{\infty}p(\mathbf{y}_k\,|\,\mathbf{x}_k)\,\tilde{p}^{j-}(\mathbf{x}_k)d\mathbf{x}_k}{\int_{-\infty}^{\infty}\hat{p}(\mathbf{y}_k\,|\,\mathbf{x}_k)\,\tilde{p}^{j-}(\mathbf{x}_k)d\mathbf{x}_k}\end{aligned} \tag{3.43}$$

As shown in Appendix D, both terms are bounded by linear functions of the covariance trace. If all the traces are less than σ_L, the componential divergence will be lower than $M\times th_L$ for some finite factor M. Then, applying Pinsker's inequality to Eq. (3.42) gives:

$$\begin{aligned}&\left\|\Gamma_A\left(\tilde{p}^-\left(\mathbf{x}_k\right)\right)-\Gamma\left(\tilde{p}^-\left(\mathbf{x}_k\right)\right)\right\| \le \sqrt{2D_{KL}\left(\Gamma_A\left(\tilde{p}^-\left(\mathbf{x}_k\right)\right)\|\,\Gamma\left(\tilde{p}^-\left(\mathbf{x}_k\right)\right)\right)}\\&\le \sqrt{2\sum_{i=1}^{N_k}\hat{w}_k^i\left(M\times th_L\right)}=\sqrt{2M\times th_L}\end{aligned} \tag{3.44}$$

Combining Eq. (3.39), (3.41) and (3.44), the L^1 norm error in Lemma 4 can be established. Q.E.D.

Lemma 5. Assume $\frac{N_k-N_{k-1}}{P-1}\sqrt{2\tilde{D}_{KL}}+\sqrt{2th_D}+\delta_{\varepsilon,k}+\sqrt{2M\times th_L}\le\Delta$ for some $\Delta<\infty$ in all time steps, then given that the condition in Eq. (3.37) holds, the L^1 norm error of estimated posterior PDF at k has $\delta_k\le C\rho^k\delta_0+\frac{C\rho+1}{1-\rho}\Delta$.

Proof: For convenience, redefine the operator Γ as the mapping of L^1 error through exact Bayesian update, and thus it satisfies $\Gamma^k\left(\delta_0\right)\le C\rho^k\delta_0$ and $\Gamma\left(\delta_1+\delta_2\right)\le C\rho\delta_1+C\rho\delta_2$. Combining the prior error in Eq. (3.29) and posterior error in Eq. (3.38) recursively gives:

$$\begin{aligned}&\delta_k\le\Gamma\left(\delta_{k-1}+\frac{N_k-N_{k-1}}{P-1}\sqrt{2\tilde{D}_{KL}}+\sqrt{2th_D}+\delta_{\varepsilon,k}\right)+\sqrt{2M\times th_L}\\&\le\Gamma\left(\Gamma\left(\delta_{k-2}+\frac{N_{k-1}-N_{k-2}}{P-1}\sqrt{2\tilde{D}_{KL}}+\sqrt{2th_D}+\delta_{\varepsilon,k-1}\right)+\Delta\right)+\Delta\\&\cdots\le\Gamma\left(\Gamma\left(\ldots\Gamma\left(\delta_0+\frac{N_1-N_0}{P-1}\sqrt{2\tilde{D}_{KL}}+\sqrt{2th_D}+\delta_{\varepsilon,1}\right)+\Delta\ldots\right)+\Delta\right)+\Delta\\&\le C\rho^k\delta_0+\sum_{i=1}^{k}C\rho^i\Delta+\Delta\le C\rho^k\delta_0+\frac{C\rho+1}{1-\rho}\Delta\end{aligned} \tag{3.45}$$

where N_k-N_{k-1} is the total increase of mixture size from time step $k-1$ to k. Q.E.D.

The Lemmas 1 and 3 proved that the refinement scheme could reach any small thresholds th_D and th_L as the trace of covariance matrix shrinks, with a square-root rate of convergence for the former and a linear rate for the latter. And then the Lemmas 2, 4 and 5 established the boundedness of the L^1 error of estimated state PDF. Although under what conditions the exponential forgetting assumption in Eq. (3.37) holds is still an active area of research [78], many dynamic systems in practice do exhibit the "forgetting initial condition" behavior under Bayesian filtering [222]. Under this forgetting assumption, the bound of state PDF error mainly depends on the magnitude of Δ in Lemma 5, which could be minimized by choosing proper neural network size to approximate the state equation and appropriate thresholds to refine the Gaussian mixture PDFs.

Remarks: 1) The thresholds th_D and th_L should not be selected lower than the KL divergence

of the adopted splitting scheme, otherwise, the gain of reducing nonlinear distortion error may not compensate the loss caused by splitting misrepresentation. 2) Since the new components from splitting inherit singular values and vectors from the original component, the SVD doesn't need to be repeated. Therefore, the AGMF that combines the linearization in EKF and sigma points in UKF to process each component has a computational complexity O(EKF+UKF)$\times N_{k-1} < O$(AGMF) $< O$(EKF+UKF)$\times N_k$, given that the mixture size is increased from N_{k-1} to N_k at step k. 3) The increased computational cost in AGMF is to handle the high nonlinearity and high magnitude of uncertainty in the system. If the system is only weakly nonlinear such that the divergences defined in Eq. (3.17) and (3.24) are inherently small without any splitting, then trivial improvement can be expected from the AGMF. Also, if the magnitude of state uncertainty is so small that the inverse $\mathbf{\Sigma}_k^{-1}$ is overwhelming compared to a feasible FFNN error $\mathbf{\Sigma}_\varepsilon$, then the δ_ε term will be nonnegligible and approximating the state equation with an FFNN may not help to improve filter fidelity. In these scenarios, the UKF or other non-mixture filters would be more efficient choices. 4) Though the AGMF is derived for Gaussian noises, it can be extended to non-Gaussian noises provided that the noise can be characterized as a Gaussian mixture. Then, the $p(\mathbf{x}_a)$ in Eq. (3.15) will have $N_{k-1} \times N_w$ components by combining the state PDF with a N_w-component noise $\mathbf{w}_k$, and the measurement likelihood will also need to be assessed for each component in $\mathbf{v}_k$. Consequently, the Gaussian mixture reduction will be more critically needed to regulate the exponential growth of mixture size in addition to the increase due to splitting.

3.3 Application Examples

In this section, to evaluate the proposed AGMF, three numerical examples are presented, in which the AGMF is compared with the widely used nonlinear Kalman filters, particle filters and other latest filters in terms of state estimation accuracy and computational efficiency.

3.3.1 Example I

The first example focuses on testing the new Bayesian update scheme with Gaussian mixture refinement, given that the uncertainty propagation scheme for state prediction has been tested on various neural network examples in Chapter 2. Using a range function $g(\mathbf{x}) = \sqrt{\mathbf{x}^T\mathbf{x}}$, the measurement equation in this example is $y = \sqrt{\mathbf{x}^T\mathbf{x}} + v$, where $\mathbf{x}=[x_1, x_2]^T$ is the state vector and v

$\sim\mathcal{N}(0, 0.02)$ is Gaussian white noise. As a replica of Example II in [62], the prior PDF of $\mathbf{x}$ was set to be $p^{-}(\mathbf{x})\sim\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ with $\boldsymbol{\mu}=[-3, 0]^T$ and $\boldsymbol{\Sigma}=\text{diag}[4, 4]$. Then assume that a measurement $y=1$ was received, the posterior PDFs estimated by different filters were compared.

The true posterior PDF was evaluated numerically on a grid of points in the state plane and its mean value was used as the true state location. The AGMF was implemented with a maximum mixture size $N_{\max}=200$ and a splitting threshold $th_L=0.001$. For comparison, an EKF, a UKF with UT parameters $\alpha=0.001$, $\beta=2$, $\kappa=0$, a sequential importance resampling (SIR) particle filter (PF) with 2.5×10^4 particles, and a state-of-the-art GMF, i.e., the blob filter in [80], were also applied. For blob filter, 1000 Gaussian components were used and the lower bound of the information matrix was chosen to be $R_{\min}=\text{diag}([5, 5])$ such that the resampled covariance would be bounded by $\boldsymbol{\Sigma}\leq R_{\min}{}^{-T}R_{\min}{}^{-1}$. The performances of these filters were compared in Table 3-1 in terms of the error of estimated mean state locations, the KL divergence between estimated and true posterior PDFs, and the mean computation time. As can be seen, the EKF and UKF failed to estimate the state accurately because of the nonlinear effect caused by the high magnitude of prior uncertainty. In contrast, the AGMF achieved the best accuracy while its computation time was similar to those of the PF and blob filter. Also, as shown in Figure 3-1, through the adaptive splitting of Gaussian prior PDF, the AGMF accurately predicted the volcano-shaped posterior PDF. Though the accuracies of PF and blob filter may be improved to the same level as the AGMF by increasing the number of samples, their computation times will then be considerably longer. Therefore, compared with these two filters, the AGMF can attain the best accuracy under the same level of computational cost, or has the best computational efficiency to attain the same level of accuracy.

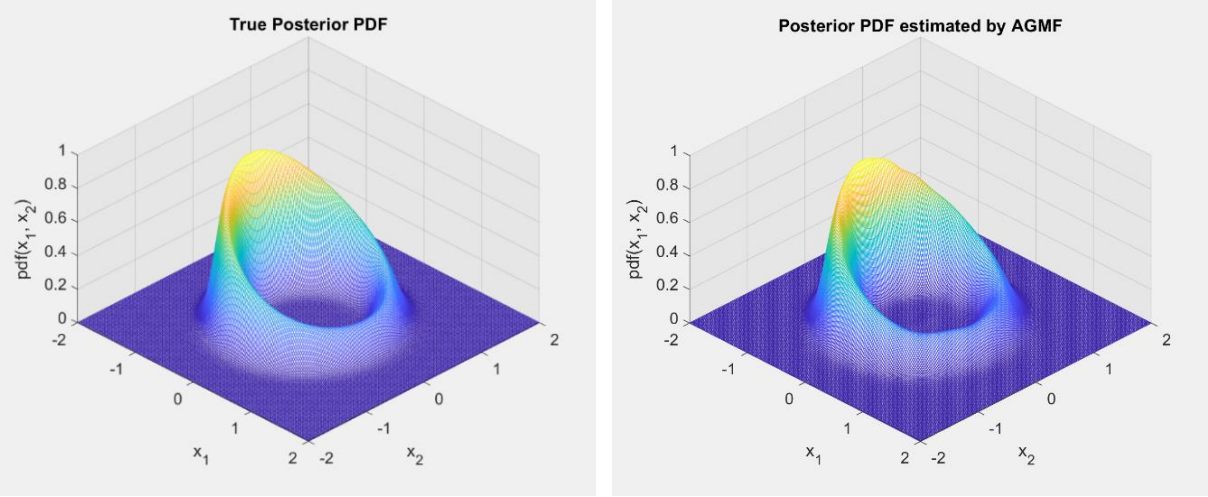


Figure 3-1 Comparison of true and estimated posterior PDFs for Example I

Table 3-1 Comparison of filter performance for Example I

	Error of mean	*KL divergence*	*Computation time*
EKF	0.641	20.64	0.56ms
UKF	0.458	1.708	0.53ms
PF	8.4×10^{-3}	0.017	55.5ms
Blob filter	0.096	0.099	62.8ms
AGMF	4.5×10^{-4}	0.026	59.3ms

3.3.2 Example II

The second example considers a complete state estimation problem for a nonlinear dynamic system with the following model [78]:

$$\begin{aligned} x_{k+1} &= \frac{x_k}{2} + \frac{25x_k}{1+x_k^{\,2}} + 8\cos\left(1.2k\right) + w_k \\ y_k &= \frac{x_k^{\,2}}{20} + v_k \end{aligned} \tag{3.46}$$

where the noises are assumed to be $w_k \sim \mathcal{N}(0,10)$ and $v_k \sim \mathcal{N}(0,1)$. The time-independent part of this state equation has two locally attractive equilibrium points at ±7 while the cosine term excites the jumping between these two modes, and thus the system may exhibit a bimodal state distribution. Also, the measurement function only contains the magnitude of x_k but not the sign, which makes it difficult for a filter to discern the mode. For this problem, an FFNN with 5 hidden nodes was trained to approximate the state equation, and then the AGMF was implemented. An EKF, a UKF with the same parameters from [78], a PF based on SIR, and a blob filter were also simulated on

this system for 52 time steps with an initial state uncertainty $p(x_0)=\mathcal{N}(0, 2)$. The parameters used by these filters are summarized in Table 3-2.

The filters are compared in terms of the root mean square error (RMSE) and computation time averaged over 100 Monte Carlo runs in Table 3-2 and Figure 3-2. It can be seen that the AGMF outperformed the EKF and UKF and achieved an RMSE similar to those of the PF and blob filter. Due to the low dimensionality of this problem, the PF could use fewer particles and hence had a better computational efficiency. In Figure 3-3, the state PDFs predicted by different filters at two representative cases are compared. The true PDF was estimated by the histogram extracted from a PF with 50000 particles, and the PDF of the 500-particle PF in Table 3-2 was obtained in the same way. As is shown, the EKF yielded too narrow Gaussians at local modes while the UKF produced too wide Gaussians to capture the global variance. The AGMF accurately predicted the bimodal PDFs with the best agreement with ground truth, while the PDFs of the PF and blob filter looked like ensembles of sparse impulses due to their limited sampling sizes. For the blob filter, though it had an RMSE similar to AGMF, using a narrow covariance upper bound for all components would lead to a coarse global approximation of state PDF if its number of components was not large enough while using a wider upper bound would increase the risk of local nonlinear distortion. In contrast, the AGMF was able to adjust the width of each Gaussian component adaptively and thus could achieve a better estimation of state PDF with fewer components.

Table 3-2 Comparison of filter performance on Example II

	RMSE	*Computation time*	*Filter parameters*
EKF	20.87	0.021s	
UKF	7.47	0.019s	α=1.3, β=1.5, κ=0.2
PF	4.20	0.065s	500 particles
Blob filter	4.73	0.275s	100 components, $R_{\min}$=100
AGMF	4.28	0.211s	$N_{\max}$=25, th_D=0.01, th_L=0.01

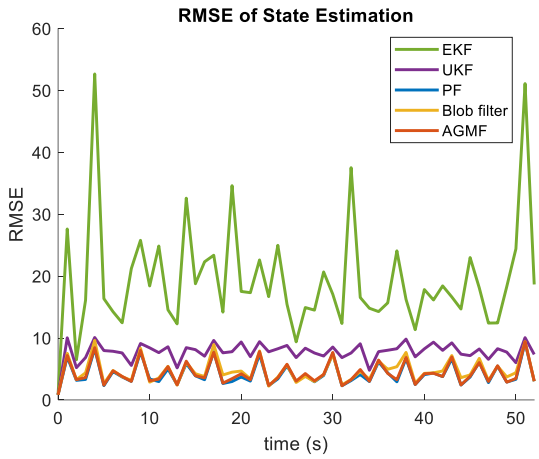


Figure 3-2 Comparison of RMSE for state estimation in Example II

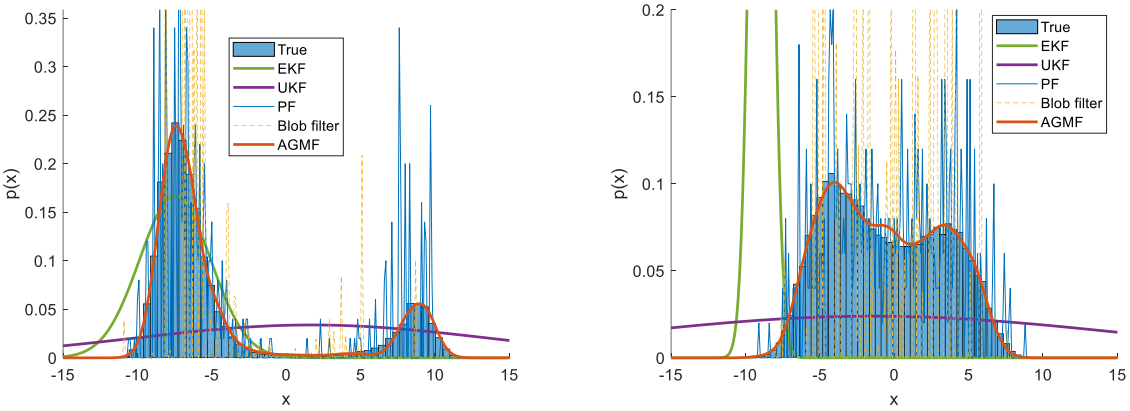


Figure 3-3 Comparison of estimated state PDFs in Example II

3.3.3 Example III

The third example explores a benchmark nonlinear filtering problem developed in [223], the "blind tricyclist" problem. Assume that a blind man is navigating a tricycle across an amusement park without knowing his own exact position. Two friends on two merry-go-rounds are shouting to him intermittently. The blind tricyclist measures the relative bearing angle between his forward direction and the direction to a friend by using his hearing to listen for shouts. He doesn't know each friend's rotation angle and rotation rate, though the center location and radius of each merry-go-round are known. The navigation involves 7 state variables $\mathbf{x}_k=[X_k, Y_k, \theta_k, \varphi_{1k}, \varphi_{2k}, \omega_{1k}, \omega_{2k}]^T$, which consists of his X-Y position coordinates, his heading angle, and the angular positions and

velocities of the two friends on their respective merry-go-rounds. More details of this problem, including the state and measurement equations, initial uncertainty, and a nominal trajectory to test filters, can be found in [223]. The duration of the test trajectory is 141 seconds, the sampling interval is 0.5-seconds, and the measurement with respect to each friend occurs every 2 seconds. An FFNN with 20 hidden nodes was trained to approximate the state equation and then the AGMF was applied along with an EKF, a UKF, a PF and two blob filters, whose parameters are summarized in Table 3-3. The lower bounds of information matrix for blob filters were adopted from [80] as $R_{\min}$ = diag[1/2.6; 1/2.6; 1/1.04; 1/0.3467; 1/0.4; 1/2000; 1/2000].

The filters were tested for 100 Monte Carlo runs with randomly drawn start positions. The RMSEs of the estimated terminal positions (at t=141 s) are compared in Table 3-3, and the RMSEs along the trajectory are compared in Figure 3-4. In addition, in Figure 3-5, the trajectories estimated by different filters are plotted and compared with the true trajectory for one simulation case. Since a fairly large initial uncertainty was used, the impact of nonlinearity was significant for the filters. As a result, the RMSEs of the EKF and UKF were not satisfactory, though they had the shortest computation time. Even the PF had a sub-optimal performance on this problem, as also observed in [80]. The AGMF and two blob filters exhibited reasonable accuracies. As shown in Figure 3-5, though these three filters deviated at the beginning, they all converged to the true trajectory after some time while the other filters were still confused, which means these Gaussian-mixture-based filters are more resistant to the impact of nonlinearity. The RMSE of the AGMF was almost the same as the blob filter with 7000 components, while its computation time was only 8% of the latter. The blob filter with 700 components had a similar computation time as the AGMF, while its RMSE was moderately larger. Therefore, it can be concluded that among all the filters tested, the AGMF made the best compromise between filter accuracy and computational efficiency on this seven-dimensional highly nonlinear problem with a high magnitude of uncertainty.

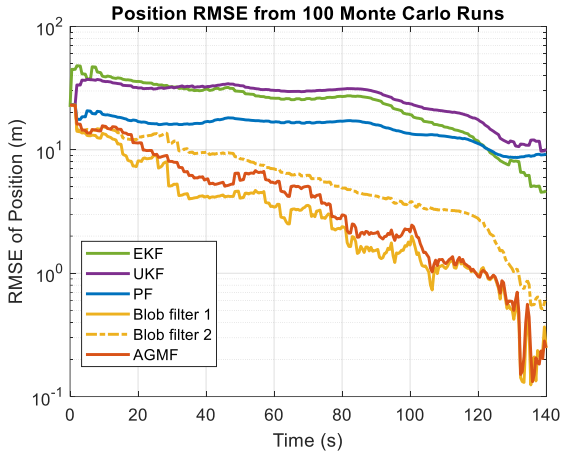


Figure 3-4 Comparison of RMSE for state estimation in Example III

Table 3-3 Comparison of filter performance on Example III

	RMSE of position	*Computation time*	*Filter parameters*
EKF	4.66	0.069s	
UKF	10.14	0.140s	α=0.001, β=2, κ=0
PF	9.30	69.80s	10000 particles
Blob filter-1	0.247	317.2s	7000 components
Blob filter-2	0.630	24.29s	700 components
AGMF	0.250	24.63s	$N_{\max}$=200, th_D=0.01, th_L=0.01

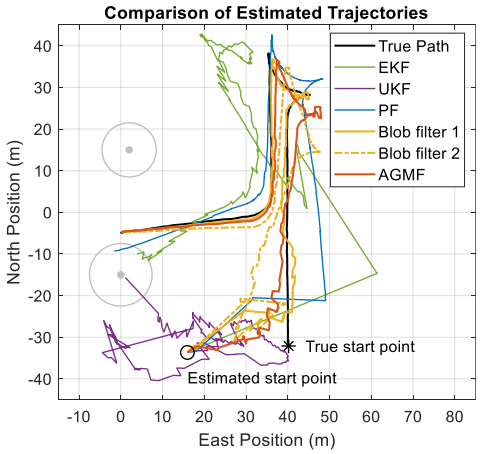


Figure 3-5 Comparison of estimated tricyclist trajectories

3.4 Summary

This chapter investigates the probabilistic state estimation in nonlinear dynamic systems. A novel adaptive Gaussian mixture filter is developed by using Gaussian mixture models to estimate

the non-Gaussian distribution of state variables. To attain a high-fidelity global approximation of state distributions, the Gaussian mixtures are refined adaptively by splitting components based on the local severity of nonlinearity. In the state prediction stage, it is proposed to approximate the complex dynamic state equation with a feedforward neural network so that the nonlinearity can be assessed more tractably at the sigmoid function layer. A likelihood divergence criterion is also proposed to assess the nonlinearity associated with measurements in the Bayesian update stage. The splitting process triggered by these designed criteria, as it shrinks the covariance of Gaussian components, can refine the Gaussian mixtures to any desired level with quantified rates of convergence. It has also been proved that under mild conditions, the L^1 norm between the estimated state PDFs and exact Bayesian solutions is upper bounded and can be regulated to a minimal level.

The proposed filter has been evaluated on multiple challenging nonlinear filtering problems. Compared to the widely used filters (e.g., EKF, UKF, particle filter) and state-of-the-art methods in the literature (e.g., blob filter), the AGMF designed in this work provides among-the-best state estimation accuracy with a reasonable computational cost. Especially, it can estimate any non-Gaussian state distribution with a compact Gaussian mixture size and uncompromised fidelity, which makes it a promising solution for highly nonlinear state estimation applications.

4. PROBABILISTIC NEURAL NETWORK FOR UNCERTAINTY PREDICTION

In this chapter, a Gaussian mixture probabilistic neural network (GM-PNN) with adaptive Gaussian mixture refinement is proposed. The network's inputs, activation states, parameters and outputs are all treated probabilistically as Gaussian mixtures, which enables the characterization of arbitrary probability distributions. The Gaussian mixtures are refined before each nonlinear activation layer using the adaptive Gaussian mixture scheme from Chapter 2 to minimize their distortions. The refinement scheme is also extended to ensure the linear transformation fidelity with probabilistic weights. With this Gaussian mixture scheme, the predictive distributions that combine the input uncertainties, nonlinear effect of the network, and parameter uncertainties can be obtained directly and analytically without sampling or integration. The derivatives of all the probabilistic parameters are derived analytically so that the GM-PNN can be trained efficiently using any backpropagation method based on gradient descent. The GM-PNN achieved state-of-the-art performance when benchmarked against other methods on a series of public datasets. Therefore, it is a promising solution to address real-world applications subject to uncertainties.

4.1 The Gaussian Mixture Probabilistic Neural Network

This work considers the probabilistic parameterization of the multi-layer feedforward neural networks. A feedforward neural network with L layers can be written as:

$$\mathbf{y} = \mathbf{W}_L f\left(\mathbf{W}_{L-1} f\left(\ldots \mathbf{W}_1 \mathbf{x} + \mathbf{b}_1 \ldots\right) + \mathbf{b}_{L-1}\right) + \mathbf{b}_L \tag{4.1}$$

where $\mathbf{x} \in \mathbb{R}^m$ is the m-dimensional vector of input features, $\mathbf{y} \in \mathbb{R}^n$ is the n-dimensional vector of output targets, $\mathbf{W}_l \in \mathbb{R}^{k_l \times k_{l-1}}$ and $\mathbf{b}_l \in \mathbb{R}^{k_l}$ are the weight and bias of the l-th layer with k_l hidden neurons (l=1,…,L), and f is the nonlinear activation function. Also, for the l-th layer, define $\mathbf{o}_l \in \mathbb{R}^{k_l}$ as the pre-activation states and $\mathbf{a}_l \in \mathbb{R}^{k_l}$ as the post-activation states (note that the pre-activation and post-activation states are denoted as **z** and **s** in Chapters 2 and 3):

$$\begin{aligned} \mathbf{o}_l &= \mathbf{W}_l \mathbf{a}_{l-1} + \mathbf{b}_l \\ \mathbf{a}_l &= f\left(\mathbf{o}_l\right) \end{aligned} \tag{4.2}$$

with $\mathbf{a}_0 = \mathbf{x}$ and $\mathbf{o}_L = \mathbf{y}$ at the input and output layers. For the GM-PNN studied in this work, the

probability density functions (PDF) of all its parameters are assumed as Gaussian mixtures, which enable the modeling of arbitrary multimodal non-Gaussian distributions:

$$p(\mathbf{W}_l, \mathbf{b}_l) = \sum_{i=1}^{N_{\mathbf{W}(l)}} w^i_{\mathbf{W}(l)} \mathcal{N}\left(\mathbf{W}_l \mid \mathbf{M}^i_{\mathbf{W}(l)}, \boldsymbol{\Sigma}^i_{\mathbf{W}(l)}\right) \mathcal{N}\left(\mathbf{b}_l \mid \boldsymbol{\mu}^i_{\mathbf{b}(l)}, \boldsymbol{\Sigma}^i_{\mathbf{b}(l)}\right) \tag{4.3}$$

where the weight and bias in the same layer share the same mixture size $N_{\mathbf{W}(l)}$ and component weights $w^i_{\mathbf{W}(l)}$. For the bias vector $\mathbf{b}_l$, $\boldsymbol{\mu}^i_{\mathbf{b}(l)} \in \mathbb{R}^{k_l}$ and $\boldsymbol{\Sigma}^i_{\mathbf{b}(l)} \in \mathbb{R}^{k_l \times k_l}$ denote its mean vector and covariance matrix in the *i*-th component, respectively, as for an ordinary multivariate Gaussian PDF. Then it is worth noting that for the weight matrix $\mathbf{W}_l$, $\mathbf{M}^i_{\mathbf{W}(l)} \in \mathbb{R}^{k_l \times k_{l-1}}$ and $\boldsymbol{\Sigma}^i_{\mathbf{W}(l)} \in \mathbb{R}^{k_l \times k_{l-1}}$ are matrices with the same size as $\mathbf{W}_l$ with each element denoting the mean and variance of each weighting factor at the corresponding position. Though it is assumed that the weighting factors are uncorrelated, the multiplication operation in Eq. (4.2) will introduce correlation for the states in $\mathbf{o}_l$, even if the covariance matrices of $\mathbf{a}_{l-1}$ and $\mathbf{b}_l$ are also diagonal.

4.1.1 Linear Transformation in GM-PNN

To infer the predictive distribution layer-wisely through the GM-PNN, assume that the PDF of activation states from the previous layer is $p(\mathbf{a}_{l-1}) = \sum_{i=1}^{N_{\mathbf{a}(l-1)}} w^i_{\mathbf{a}(l-1)} \mathcal{N}\left(\mathbf{a}_{l-1} \mid \boldsymbol{\mu}^i_{\mathbf{a}(l-1)}, \boldsymbol{\Sigma}^i_{\mathbf{a}(l-1)}\right)$, then the exact PDF of $\mathbf{o}_l$ at the current layer should be:

$$p(\mathbf{o}_l) = \sum_{i,j=1}^{N_{\mathbf{a}(l-1)} \times N_{\mathbf{W}(l)}} w^i_{\mathbf{a}(l-1)} w^j_{\mathbf{W}(l)} \int \begin{array}{l} \mathcal{N}\left(\mathbf{o}_l \mid \mathbf{W}_l \boldsymbol{\mu}^i_{\mathbf{a}(l-1)} + \mathbf{b}_l, \mathbf{W}_l \boldsymbol{\Sigma}^i_{\mathbf{a}(l-1)} \mathbf{W}_l^T\right) \\ \times \mathcal{N}\left(\mathbf{M}^j_{\mathbf{W}(l)}, \boldsymbol{\Sigma}^j_{\mathbf{W}(l)}\right) \mathcal{N}\left(\boldsymbol{\mu}^j_{\mathbf{b}(l)}, \boldsymbol{\Sigma}^j_{\mathbf{b}(l)}\right) d\mathbf{W}_l d\mathbf{b}_l \end{array} \tag{4.4}$$

Although this transformation from $\mathbf{a}_{l-1}$ to $\mathbf{o}_l$ is linear, the integration is in general intractable. This is mainly due to the multiplication of probabilistic state $\mathbf{a}_{l-1}$ with probabilistic weight $\mathbf{W}_l$, given that the product of two Gaussian random variables is non-Gaussian. Hence, rather than pursuing exact integration, the moment matching solution is used instead to approximate the integration for each pair of components [48] and maintain $p(\mathbf{o}_l)$ in a tractable Gaussian mixture form:

$$p(\mathbf{o}_l) = \sum_{h=1}^{N_{\mathbf{o}(l)}} w_{\mathbf{o}(l)}^h p^h(\mathbf{o}_l) = \sum_{h=1}^{N_{\mathbf{o}(l)}} w_{\mathbf{o}(l)}^h \mathcal{N}\left(\mathbf{o}_l \mid \boldsymbol{\mu}_{\mathbf{o}(l)}^h, \boldsymbol{\Sigma}_{\mathbf{o}(l)}^h\right)$$

with

$$\begin{aligned}
&N_{\mathbf{o}(l)} = N_{\mathbf{a}(l-1)} \times N_{\mathbf{W}(l)} \quad i = 1, \ldots, N_{\mathbf{a}(l-1)} \\
&w_{\mathbf{o}(l)}^h = w_{\mathbf{a}(l-1)}^i w_{\mathbf{W}(l)}^j \qquad j = 1, \ldots, N_{\mathbf{W}(l)} \\
&\boldsymbol{\mu}_{\mathbf{o}(l)}^h = \mathbf{M}_{\mathbf{W}(l)}^j \boldsymbol{\mu}_{\mathbf{a}(l-1)}^i + \boldsymbol{\mu}_{\mathbf{b}(l)}^j \\
&\boldsymbol{\Sigma}_{\mathbf{o}(l)}^h = \operatorname{diag}\left\{\boldsymbol{\Sigma}_{\mathbf{W}(l)}^j \mathbf{s}_{\mathbf{a}(l-1)}^i\right\} + \operatorname{diag}\left\{\boldsymbol{\Sigma}_{\mathbf{W}(l)}^j \left(\boldsymbol{\mu}_{\mathbf{a}(l-1)}^i \circ \boldsymbol{\mu}_{\mathbf{a}(l-1)}^i\right)\right\} \\
&\qquad + \mathbf{M}_{\mathbf{W}(l)}^j \boldsymbol{\Sigma}_{\mathbf{a}(l-1)}^i \mathbf{M}_{\mathbf{W}(l)}^{j\,T} + \boldsymbol{\Sigma}_{\mathbf{b}(l)}^j
\end{aligned} \tag{4.5}$$

where $\circ$ denotes element-wise product, diag denotes the operation of converting a vector to a diagonal matrix and $\mathbf{s}_{\mathbf{a}(l-1)}^i$ is the vector of diagonal elements of $\boldsymbol{\Sigma}_{\mathbf{a}(l-1)}^i$. The four terms in $\boldsymbol{\Sigma}_{\mathbf{o}(l)}^h$ correspond to the covariance due to the coupling between $\boldsymbol{\Sigma}_{\mathbf{a}(l-1)}^i$ and $\boldsymbol{\Sigma}_{\mathbf{W}(l)}^j$, the covariance when only the weight $\mathbf{W}_l$ is probabilistic, the covariance when only the state $\mathbf{a}_{l-1}$ is probabilistic, and the covariance of bias $\mathbf{b}_l$, respectively.

This moment matching solution may have approximation errors. But, if $\boldsymbol{\Sigma}_{\mathbf{a}(l-1)}^i$ is so narrow that $p^i(\mathbf{a}_{l-1})$ degenerates into a Dirac delta function, then the integrals in Eq. (4.4) will converge to the solution in Eq. (4.5). A relaxed condition for a componential density in $\mathbf{o}_l$ to be Gaussian is the coupling term $\boldsymbol{\Sigma}_{\mathbf{W}(l)}\mathbf{s}_{\mathbf{a}(l-1)}$ is zero for that component. The proof can be straightforward by considering the single Gaussian PDF case: given the weight matrix $\mathbf{W}_l = \left[W_{l,i,j}\right]$ (i = 1, …, k_l, j = 1, …, k_{l-1}), a state in $\mathbf{o}_l = \left[o_{l,1}, \ldots, o_{l,k_l}\right]^T$ has $o_{l,i} = b_{l,i} + \sum_{j=1}^{k_{l-1}} W_{l,i,j} a_{l-1,j}$. The coupling term for $o_{l,i}$ equals zeros means $\sum_{j=1}^{k_{l-1}} \operatorname{var}\left(W_{l,i,j}\right)\operatorname{var}\left(a_{l-1,j}\right) = 0$. Since all the variances are non-negative, either $\operatorname{var}\left(W_{l,i,j}\right)$ or $\operatorname{var}\left(a_{l-1,j}\right)$ must be 0, which implies one of $W_{l,i,j}$ or $a_{l-1,j}$ for the same index j is deterministic such that their product is a deterministic weight (or state) times a probabilistic state (or weight). In such a case, $o_{l,i}$ equals the sum of k_{l-1} Gaussian variables scaled by deterministic factors and will have a Gaussian distribution, as the scaling and summation of Gaussian variables is still Gaussian. Therefore, the smaller the coupling term is, the better the moment matching

Gaussians can approximate the exact solution. Based on the above discussion of the condition for a density in $\mathbf{o}_l$ to be Gaussian and following the covariance equation in last line in Eq. (4.5), a measure of the covariance coupling, denoted as Cov_{CP}, between $p^i(\mathbf{a}_{l-1})$ and $p^j(\mathbf{W}_l)$ is defined as:

$$\begin{aligned} D_{CP}\left(p^i(\mathbf{a}_{l-1}) \| p^j(\mathbf{W}_l)\right) = & \left(\boldsymbol{\Sigma}^j_{\mathbf{W}(l)}\mathbf{s}^i_{\mathbf{a}(l-1)}\right)^T \\ & \times\left(\boldsymbol{\Sigma}^j_{\mathbf{W}(l)}\mathbf{s}^i_{\mathbf{a}(l-1)} + \boldsymbol{\Sigma}^j_{\mathbf{W}(l)}\left(\boldsymbol{\mu}^i_{\mathbf{a}(l-1)} \circ \boldsymbol{\mu}^i_{\mathbf{a}(l-1)}\right) + \mathbf{s}^i_{\mathbf{o}(l)}\right)^{-1} \end{aligned} \tag{4.6}$$

where $\mathbf{s}^i_{\mathbf{o}(l)}$ is the vector of diagonal elements of $\mathbf{M}^j_{\mathbf{W}(l)}\boldsymbol{\Sigma}^i_{\mathbf{a}(l-1)}{\mathbf{M}^j_{\mathbf{W}(l)}}^T$ and the superscript -1 denotes the element-wise reciprocal operation on a column vector. How to use this quantity to ensure the fidelity of predictive distribution inference will be discussed in the next subsection.

4.1.2 Nonlinear Transformation in GM-PNN

Given a pre-activation Gaussian mixture PDF $p(\mathbf{o}_l)$, the nonlinear activation function *f* will distort each of its Gaussian components such that the components in the PDF $p(\mathbf{a}_l)$ of post-activation state $\mathbf{a}_l = f(\mathbf{o}_l)$ will no longer be Gaussian. If the activation function *f* is piecewise differentiable and monotonically increasing, like those listed in Table 2-1, then $p(\mathbf{a}_l)$ have an analytical solution. However, as discussed in Section 2.2.1, this exact solution is no longer Gaussian and thus may not be tractable in the following layers. For this reason, the approximate solution by linearizing *f* at the center of each Gaussian component is often used, which will converge to the exact solution if the covariance $\boldsymbol{\Sigma}^i_{\mathbf{o}(l)}$ is infinitesimal [57].

The adaptive Gaussian mixture scheme developed in Chapter 2 can be applied to assure the fidelity of PDF propagation in the nonlinear transformation of *f*. The quality of a linearization-based density is assessed by its KL divergence with respect to its exact solution counterpart $D_{KL}\left(p^i(\mathbf{a}_l) \| \hat{p}^i(\mathbf{a}_l)\right)$, as derived in Appendix A. The evaluation of this KL divergence based on unscented transform and SVD of covariance matrices has been elaborated in Section 2.2.1. If the KL divergence of a component *i* exceeds a threshold, it means the linearization-based solution will yield a considerable error in propagating $p^i(\mathbf{o}_l)$. Therefore, this component will be refined by splitting it into a set of sub-components with narrower covariance.

Before proceeding to the refinement, let's revisit the linear transformation in Section 4.1.1. As has been mentioned, the moment matching solution in Eq. (4.5) will be close to the exact solution in Eq. (4.4) if the covariance coupling term is small. The coupling term can be minimized if $\mathbf{W}_l$ is deterministic (i.e., non-probabilistic network, which is outside the scope of this work) or if the components in $p(\mathbf{a}_{l-1})$ are narrow. Therefore, similar to the nonlinear transformation, the fidelity of linear transformation could also be improved by splitting components to narrow their covariance matrices. Besides, to save computational cost, it is preferred that the splittings for linear and nonlinear transformations can be performed together so that the SVD of covariance matrices required by splitting only needs to be performed once. For a pre-activation density $p^i(\mathbf{o}_l)$, the total covariance coupling between its approximate post-activation density $\hat{p}^i(\mathbf{a}_l)$ and the next layer's weight $\mathbf{W}_{l+1} \sim \sum_{i=1}^{N_{\mathbf{W}(l+1)}} w^i_{\mathbf{W}(l+1)} p^j(\mathbf{W}_{l+1})$ can be assessed by the weighted sum of the componential couplings defined in Eq. (4.6):

$$Cov_{CP}\left(\hat{p}^i(\mathbf{a}_l) \,\|\, p(\mathbf{W}_{l+1})\right) = \sum_{i=1}^{N_{\mathbf{W}(l+1)}} w^i_{\mathbf{W}(l+1)} D_{CP}\left(\hat{p}^i(\mathbf{a}_l) \,\|\, p^j(\mathbf{W}_{l+1})\right) \tag{4.7}$$

In addition to the KL divergence assessment using UT to ensure the nonlinear transformation fidelity at layer l, this coupling assessment by looking ahead at the next layer's weight could trigger refinement to ensure the linear transformation fidelity at layer l+1. If the covariance $\mathbf{\Sigma}^i_{\mathbf{o}(l)}$ is refined such that $\mathbf{\Sigma}^i_{\mathbf{a}(l)} = \mathbf{A}^i \mathbf{\Sigma}^i_{\mathbf{o}(l)} \mathbf{A}^{iT}$ is sufficiently narrow, then the magnitude of $\mathbf{\Sigma}^j_{\mathbf{W}(l+1)} \mathbf{s}^i_{\mathbf{a}(l)}$ can be much lower than the $\mathbf{\Sigma}^j_{\mathbf{W}(l+1)} \left(\boldsymbol{\mu}^i_{\mathbf{a}(l)} \circ \boldsymbol{\mu}^i_{\mathbf{a}(l)}\right) + \mathbf{s}^i_{\mathbf{o}(l)}$ in Eq. (4.6), because the latter also depends on the mean magnitude of $\mathbf{a}_l$ and $\mathbf{W}_{l+1}$, and thus will not be infinitesimal. Thereby, the covariance coupling is negligible and the PDF of $\mathbf{o}_{l+1}$ after the linear transformation at layer l+1 could be closely approximated as a Gaussian mixture using Eq. (4.5).

The KL divergence and covariance coupling will be assessed for every component in $p(\mathbf{o}_l)$. For a component $w^i_{\mathbf{o}(l)} \mathcal{N}\left(\mathbf{o}_l \,|\, \boldsymbol{\mu}^i_{\mathbf{o}(l)}, \mathbf{\Sigma}^i_{\mathbf{o}(l)}\right)$, if either of these two quantities exceeds their respective thresholds, it will be split into a sub-mixture of P components along its singular vector with the largest singular value. The splitting operation using a scheme selected from Table 2-2 is the same as Eq. (2.18) and Eq. (3.19). Assume that by repeating the splitting process until no component

triggers new splitting, the PDF of $\mathbf{o}_l$ is refined as $\tilde{p}(\mathbf{o}_l) = \sum_{i=1}^{\tilde{N}_{\mathbf{o}(l)}} \tilde{w}_{\mathbf{o}(l)}^{i} \mathcal{N}\left(\mathbf{o}_l \mid \tilde{\boldsymbol{\mu}}_{\mathbf{o}(l)}^{i}, \tilde{\boldsymbol{\Sigma}}_{\mathbf{o}(l)}^{i}\right)$ (note that the mixture size increases to $\tilde{N}_{\mathbf{o}(l)}$), then this refined Gaussian mixture PDF can be propagated through the activation function *f* using UT, as in Eq. (2.19).

The obtained post-activation PDF $p(\mathbf{a}_l)$ is then sent to Eq. (4.5) again to perform the linear transformation at layer *l*+1. The advantage of this unitive refinement process by extending the one in Section 2.2 is that by considering both the criteria of KL divergence and covariance coupling, it not only minimizes the nonlinear transformation distortion of $p(\mathbf{a}_l)$ at layer *l*, but also ensures the linear transformation fidelity of $p(\mathbf{o}_{l+1})$ at layer *l*+1. By performing the linear and nonlinear transformations layer by layer using this adaptive Gaussian mixture scheme until the last layer (*l*=*L*), the PDF of output $p(\mathbf{y}) = p(\mathbf{o}_L)$ can be predicted.

To infer predictive distributions, the existing BNNs typically have to presume a Gaussian likelihood function with a problem-specific model precision hyperparameter and then integrate the likelihood over the posterior distribution of weights [90][93]. The integration is intractable in most cases and thus approximate inference methods like Monte Carlo sampling have to be used. In contrast, the predictive distribution of output in our GM-PNN can be inferred analytically by propagating Gaussian mixture probabilities forward layer by layer with all the uncertainties of input, states and parameters considered. The direct prediction of high-fidelity output distributions without sampling or evaluation of integration is the main advantage of the proposed GM-PNN over other probabilistic networks like BNNs. The predictive distribution inference algorithm of GM-PNN is summarized in Algorithm III.

Lastly, since the refinement and linear transformation will increase the number of Gaussian components in $p(\mathbf{o}_l)$ and $p(\mathbf{a}_l)$ exponentially over layers while the actual PDF is not going to be infinitely complex, the mixture size of $p(\mathbf{W}_l)$ should be limited when defining the structure of GM-PNN ($N_{\mathbf{W}(l)}$ = 2 or 3 should be adequate to model any multimodal distribution of parameters) and a Gaussian mixture reduction step, like the one in Section 2.2.4, can be used to merge components and reduce the computational complexity for very deep networks.

Algorithm III: inference of predictive distribution in GM-PNN

Given the input with Gaussian mixture uncertainty $p(\mathbf{x}) = \sum_{i=1}^{N_\mathbf{x}} w_\mathbf{x}^i \mathcal{N}\left(\boldsymbol{\mu}_\mathbf{x}^i, \boldsymbol{\Sigma}_\mathbf{x}^i\right)$ (or $p(\mathbf{x}) = \mathcal{N}(\mathbf{x}, \mathbf{0})$ for a crisp-value input)

(1) Let $p(\mathbf{a}_0) = p(\mathbf{x})$ and use Eq. (4.5) to predict $p(\mathbf{o}_1)$, set l=1.
(2) For each component in $p(\mathbf{o}_l)$, evaluate the KL divergence D_{KL} and compare with the threshold th_{KL}.
(3) Evaluate the covariance coupling Cov_{CP} and compare with the threshold th_{CP}
(4) If $D_{KL} > th_{KL}$ or $Cov_{CP} > th_{CP}$ split this component
(5) Repeat steps (2)-(4) until there is no new splitting triggered.
(6) Use UT to predict $p(\mathbf{a}_l)$ from the nonlinear transformation through f.
(7) Use Eq. (4.5) to predict $p(\mathbf{o}_{l+1})$ from the linear transformation with $\mathbf{W}_{l+1}$ and $\mathbf{b}_{l+1}$.
(8) Set l=l+1, repeat steps (2)-(7) until l=L−1.
(9) Extract the predictive distribution of output $p(\mathbf{y}) = p(\mathbf{o}_L)$.

4.2 Backpropagation with Parameter Uncertainty

Training a feedforward neural network as a deterministic model can be quite easy given the well-developed backpropagation techniques. In contrast, the training of probabilistic networks like BNN is difficult because the inference of posterior distributions of parameters is intractable and hence some approximate methods, like VI or EP, must be used. In this work, since the GM-PNN can infer predictive distributions analytically, it will be possible to build a backpropagation scheme in which its probabilistic parameters can be learned like in deterministic networks, instead of using a sampling-based approximation of gradients as in [16] and [91].

The structure of a GM-PNN is defined by a hyperparameter set $\theta = \left\{L, m, n, \left\{k_l, N_{\mathbf{W}(l)}\right\}_{l=1,\ldots,L}\right\}$ where L is the number of layers, m is the input dimension, n is the output dimension, k_l is the number of hidden nodes and $N_{\mathbf{W}(l)}$ is the number of components for the distribution of $\{\mathbf{W}_l, \mathbf{b}_l\}$ in layer l. Given a dataset consisting of N samples $D = \{\mathbf{X}, \mathbf{Y}\}$ with $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ and $\mathbf{Y} = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N\}$, the training of GM-PNN is to find a parameter set $\Theta = \left\{w_{\mathbf{W}(l)}^i, \mathbf{M}_{\mathbf{W}(l)}^i, \boldsymbol{\Sigma}_{\mathbf{W}(l)}^i, \boldsymbol{\mu}_{\mathbf{b}(l)}^i, \boldsymbol{\Sigma}_{\mathbf{b}(l)}^i\right\}_{\substack{i=1,\ldots N_{\mathbf{W}(l)} \\ l=1,\ldots,L}}$ that minimizes a loss function. The loss function used in this work is negative log-likelihood (NLL):

$$L(\mathbf{X},\mathbf{Y},\Theta)=-\sum_{i=1}^{N}\log p(\mathbf{y}_i\,|\,\mathbf{x}_i,\Theta) \tag{4.8}$$

where $p(\mathbf{y}_i\,|\,\mathbf{x}_i,\Theta)$ is the likelihood of measurement $\mathbf{y}_i$ under the predictive distribution of input $\mathbf{x}_i$ and parameter set Θ. The NLL loss indicates how well the model with parameter Θ describes the dataset *D* and is widely used in the maximum likelihood estimation (MLE) learning schemes.

For GM-PNN, its likelihood for each data pair $(\mathbf{x}_i,\mathbf{y}_i)$ could be readily computed from the predictive distribution generated in Section 4.1, and thus the NLL is an analytical expression of the parameter set Θ. If the derivatives of NLL with respect to each parameter in Θ are derived, then a gradient-descent method can be applied to train the GM-PNN. Following this idea, each epoch in the gradient-based training will consist of a feedforward step and a backpropagation step. In the feedforward stage, each input sample $\mathbf{x}_i$ is propagated forward through GM-PNN with the current parameter set Θ, with appropriate refinements performed and the distributions of states $\mathbf{o}_l$ and $\mathbf{a}_l$ recorded at each intermediate layer. The likelihood $p(\mathbf{y}_i\,|\,\mathbf{x}_i,\Theta)$ is acquired by substituting $\mathbf{y}_i$ into the predicted output distribution. Then in the backpropagation stage, the gradient $\nabla_\Theta L(\mathbf{X},\mathbf{Y},\Theta)$ at the current parameter set Θ is evaluated backwardly from output to input based on the recorded intermediate-layer distributions, and the parameters in Θ are updated along the opposite gradient direction. Since the feedforward inference has been detailed in Section 4.1 and the gradient-descent algorithms have been well developed in the literature [224], the remaining of this section will focus on deriving the gradients of the NLL loss function.

First, the gradients of NLL for each sample $L^j=-\log p(\mathbf{y}_j\,|\,\mathbf{x}_j,\Theta)$ with respect to the PDF at the output layer $p(\mathbf{y})=p(\mathbf{o}_L)=\sum_{i=1}^{N_{\mathbf{o}(L)}} w^i_{\mathbf{o}(L)}\mathcal{N}\left(\mathbf{o}_L\,|\,\boldsymbol{\mu}^i_{\mathbf{o}(L)},\boldsymbol{\Sigma}^i_{\mathbf{o}(L)}\right)$ can be derived as:

$$
\begin{aligned}
\frac{\partial L^j}{\partial w^i_{\mathbf{o}(L)}} &= -\frac{\mathcal{N}\left(\mathbf{y}_j \mid \boldsymbol{\mu}^i_{\mathbf{o}(L)}, \boldsymbol{\Sigma}^i_{\mathbf{o}(L)}\right)}{\sum_{i=1}^{N_{\mathbf{o}(L)}} w^i_{\mathbf{o}(L)} \mathcal{N}\left(\mathbf{y}_j \mid \boldsymbol{\mu}^i_{\mathbf{o}(L)}, \boldsymbol{\Sigma}^i_{\mathbf{o}(L)}\right)} \\
\frac{\partial L^j}{\partial \boldsymbol{\mu}^i_{\mathbf{o}(L)}} &= -\frac{w^i_{\mathbf{o}(L)} \mathcal{N}\left(\mathbf{y}_j \mid \boldsymbol{\mu}^i_{\mathbf{o}(L)}, \boldsymbol{\Sigma}^i_{\mathbf{o}(L)}\right)}{\sum_{i=1}^{N_{\mathbf{o}(L)}} w^i_{\mathbf{o}(L)} \mathcal{N}\left(\mathbf{y}_j \mid \boldsymbol{\mu}^i_{\mathbf{o}(L)}, \boldsymbol{\Sigma}^i_{\mathbf{o}(L)}\right)} \times \mathbf{t} \\
\frac{\partial L}{\partial \boldsymbol{\Sigma}^i_{\mathbf{o}(L)}} &= -\frac{0.5 w^i_{\mathbf{o}(L)} \mathcal{N}\left(\mathbf{y}_j \mid \boldsymbol{\mu}^i_{\mathbf{o}(L)}, \boldsymbol{\Sigma}^i_{\mathbf{o}(L)}\right)}{\sum_{i=1}^{N_{\mathbf{o}(L)}} w^i_{\mathbf{o}(L)} \mathcal{N}\left(\mathbf{y}_j \mid \boldsymbol{\mu}^i_{\mathbf{o}(L)}, \boldsymbol{\Sigma}^i_{\mathbf{o}(L)}\right)} \times \left(\mathbf{t}\mathbf{t}^T - {\boldsymbol{\Sigma}^i_{\mathbf{o}(L)}}^{-1}\right) \\
\text{with} \quad \mathbf{t} &= {\boldsymbol{\Sigma}^i_{\mathbf{o}(L)}}^{-1}\left(\mathbf{y}_j - \boldsymbol{\mu}^i_{\mathbf{o}(L)}\right)
\end{aligned}
\tag{4.9}
$$

Then by knowing $\left\{\frac{\partial L^j}{\partial w^i_{\mathbf{o}(l)}}, \frac{\partial L^j}{\partial \boldsymbol{\mu}^i_{\mathbf{o}(l)}}, \frac{\partial L^j}{\partial \boldsymbol{\Sigma}^i_{\mathbf{o}(l)}}\right\}$ at layer *l* and based on the transformation in Eq. (4.5), the gradients with respect to the weight $\mathbf{W}_l \sim \sum_{k=1}^{N_{\mathbf{w}(l)}} w^k_{\mathbf{w}(l)} \mathcal{N}\left(\mathbf{W}_l \mid \mathbf{M}^k_{\mathbf{w}(l)}, \boldsymbol{\Sigma}^k_{\mathbf{w}(l)}\right)$ are:

$$
\begin{aligned}
\frac{\partial L^j}{\partial w^k_{\mathbf{w}(l)}} &= \sum_{i=1}^{N_{\mathbf{a}(l-1)}} w^i_{\mathbf{a}(l-1)} \frac{\partial L^j}{\partial w^{h_{i,k}}_{\mathbf{o}(l)}} \\
\frac{\partial L^j}{\partial \mathbf{M}^k_{\mathbf{w}(l)}} &= \sum_{i=1}^{N_{\mathbf{a}(l-1)}} \left(\frac{\partial L^j}{\partial \boldsymbol{\mu}^{h_{i,k}}_{\mathbf{o}(L)}} \otimes {\boldsymbol{\mu}^i_{\mathbf{a}(l-1)}}^T + 2 \frac{\partial L^j}{\partial \boldsymbol{\Sigma}^{h_{i,k}}_{\mathbf{o}(L)}} \mathbf{M}^k_{\mathbf{w}(l)} \boldsymbol{\Sigma}^i_{\mathbf{a}(l-1)} \right) \\
\frac{\partial L^j}{\partial \boldsymbol{\Sigma}^k_{\mathbf{w}(l)}} &= \sum_{i=1}^{N_{\mathbf{a}(l-1)}} \left[\mathbf{d}^{j,h_{i,k}}_{\mathbf{o}(l)} \otimes \left(\mathbf{s}^i_{\mathbf{a}(l-1)} + \boldsymbol{\mu}^i_{\mathbf{a}(l-1)} \circ \boldsymbol{\mu}^i_{\mathbf{a}(l-1)} \right) \right]
\end{aligned}
\tag{4.10}
$$

where $\circ$ is element-wise product, $\otimes$ is the Kronecker product, $h_{i,k}$ is the index of components in $p(\mathbf{o}_l)$ that are generated by the *i*-th Gaussian component from $p(\mathbf{a}_{l-1})$ and *k*-th component from $p(\mathbf{W}_l)$. In addition, $\mathbf{s}^i_{\mathbf{a}(l-1)}$ is the vector of diagonal elements of $\boldsymbol{\Sigma}^i_{\mathbf{a}(l-1)}$ and $\mathbf{d}^{j,h_{i,k}}_{\mathbf{o}(l)}$ is the vector of diagonal elements in $\frac{\partial L^j}{\partial \boldsymbol{\Sigma}^{h_{i,k}}_{\mathbf{o}(L)}}$. Similarly, the gradients with respect to the bias $\mathbf{b}_l$ are:

$$
\begin{aligned}
\frac{\partial L^j}{\partial \boldsymbol{\mu}^k_{\mathbf{b}(l)}} &= \sum_{i=1}^{N_{\mathbf{a}(l-1)}} \frac{\partial L^j}{\partial \boldsymbol{\mu}^{h_{i,k}}_{\mathbf{o}(L)}} \\
\frac{\partial L^j}{\partial \boldsymbol{\Sigma}^k_{\mathbf{b}(l)}} &= \sum_{i=1}^{N_{\mathbf{a}(l-1)}} \frac{\partial L^j}{\partial \boldsymbol{\Sigma}^{h_{i,k}}_{\mathbf{o}(L)}}
\end{aligned}
\tag{4.11}
$$

The bias $\mathbf{b}_l$ has the same mixture size and component weights as $\mathbf{W}_l$, while the gradients about it are much easier to evaluate since the bias is only additive to the states.

Next, to move one layer back, the gradients of NLL with respect to the post-activation PDF at layer $l-1$ can be evaluated as:

$$\begin{aligned}
\frac{\partial L^j}{\partial w^i_{\mathbf{a}(l-1)}} &= \sum\nolimits_{k=1}^{N_{\mathbf{w}(l)}} w^k_{\mathbf{w}(l)} \frac{\partial L^j}{\partial w^{h_{i,k}}_{\mathbf{o}(l)}} \\
\frac{\partial L^j}{\partial \boldsymbol{\mu}^i_{\mathbf{a}(l-1)}} &= \sum\nolimits_{k=1}^{N_{\mathbf{w}(l)}} \left(\mathbf{M}^k_{\mathbf{w}(l)}{}^T \frac{\partial L^j}{\partial \boldsymbol{\mu}^{h_{i,k}}_{\mathbf{o}(L)}} + \boldsymbol{\Sigma}^k_{\mathbf{w}(l)}{}^T \mathbf{d}^{j,h_{i,k}}_{\mathbf{o}(l)} \circ 2\boldsymbol{\mu}^i_{\mathbf{a}(l-1)} \right) \\
\frac{\partial L^j}{\partial \boldsymbol{\Sigma}^i_{\mathbf{a}(l-1)}} &= \sum\nolimits_{k=1}^{N_{\mathbf{w}(l)}} \left[\mathbf{M}^k_{\mathbf{w}(l)}{}^T \frac{\partial L^j}{\partial \boldsymbol{\Sigma}^{h_{i,k}}_{\mathbf{o}(L)}} \mathbf{M}^k_{\mathbf{w}(l)} + \operatorname{diag}\left(\boldsymbol{\Sigma}^k_{\mathbf{w}(l)}{}^T \mathbf{d}^{j,h_{i,k}}_{\mathbf{o}(l)} \right) \right]
\end{aligned} \tag{4.12}$$

Further, to move through the nonlinear activation function f:

$$\begin{aligned}
\frac{\partial L^j}{\partial w^i_{\mathbf{o}(l-1)}} &= \frac{\partial L^j}{\partial w^i_{\mathbf{a}(l-1)}} \\
\frac{\partial L^j}{\partial \boldsymbol{\mu}^i_{\mathbf{o}(l-1)}} &= \mathbf{A}^{iT} \frac{\partial L^j}{\partial \boldsymbol{\mu}^i_{\mathbf{a}(l-1)}} \\
\frac{\partial L^j}{\partial \boldsymbol{\Sigma}^i_{\mathbf{o}(l-1)}} &= \mathbf{A}^{iT} \frac{\partial L^j}{\partial \boldsymbol{\Sigma}^i_{\mathbf{a}(l-1)}} \mathbf{A}^i
\end{aligned} \tag{4.13}$$

where $\mathbf{A}^i$ is the Jacobian matrix of f at $\boldsymbol{\mu}^i_{\mathbf{o}(l)}$. The above gradients are all derived based on matrix calculus. The Eq. (4.10) and (4.11) are used to update the parameters in layer l, while Eq. (4.12) and (4.13) are used to backpropagate to layer $l-1$. The gradients in Eq. (4.13) are then sent to Eq. (4.10) and (4.11) to update the parameters in layer $l-1$ repetitively until the input layer.

Based on the explicitly given gradients, any gradient-descent algorithm can be used to train the GM-PNN. In this work, the stochastic gradient descent with momentum (SGDM) is adopted, as will be presented in the next section. Though the training scheme in this section is not Bayesian as it does not explicitly involve a prior distribution of Θ, it's implicitly related to Bayesian networks because the negative log-likelihood term is also in the loss function of BNNs. While BNNs also have a KL-divergence term with respect to the prior distribution in their loss function, this term typically just acts like a regularization term to limit the model complexity and prevent overfitting [82]. For the GM-PNN, a regularization term could also be added in Eq. (4.8) to achieve the same effect as using a prior, while it does not necessarily make the training explicitly Bayesian.

4.3 Application Examples

In this section, the GM-PNN is tested on a series of benchmark datasets and compared with other state-of-the-art methods in the literature to validate its efficiency in learning from noisy data by considering uncertainties. Four popularly used datasets of regression tasks from the UCI machine learning repository, on which many machine learning models are benchmarked, were selected to test the GM-PNN. The datasets include the Boston housing dataset (506 samples with 13 features), concrete strength dataset (1030 samples with 9 features), energy efficiency dataset (768 samples with 8 features), and yacht hydrodynamics dataset (308 samples with 7 features). Three state-of-the-art machine learning models, all with an emphasis on uncertainty learning, were selected for comparison: the deep Gaussian process (Deep GP) in [50], the concrete dropout (C-Dropout) model in [93], and the generalized expectation propagation (GEP) model in [95]. All the models compared were composed of a single hidden layer and 50 hidden nodes, which means the model complexity was roughly the same for all the learning methods evaluated. The GM-PNN training was carried out using stochastic gradient descent with momentum (SGDM) with a learning rate of 0.01 and a momentum of 0.9. The splitting thresholds for both D_{KL} and D_{CP} were selected as 0.01, and the number of components in the PDFs of $\mathbf{W}_l$ and $\mathbf{b}_l$ was set to be 2 for all layers.

The datasets were randomly split into a 90% training set and a 10% test set for each round of evaluation. The splitting and evaluation were repeated 20 times for each dataset, and the average RMSE and negative log-likelihood (NLL) on test sets were compared for the selected models, as shown in Table 4-1. The GM-PNN ranked 1st three times out of the four datasets for test RMSE and two times for test NLL. On those datasets where it is not the champion, its performance was also fairly close to the best one. Therefore, the overall performance of GM-PNN in this comparison was remarkably better than other models.

Table 4-1 Average test RMSE and negative log-likelihood (NLL) for the selected models

	Avg. Test RMSE				**Avg. Test NLL**			
	Deep GP	**C-Dropout**	**GEP**	**GM-PNN**	**Deep GP**	**C-Dropout**	**GEP**	**GM-PNN**
Boston	3.02±0.20	2.65±0.17	2.96±0.13	**2.57±0.15**	**2.33±0.06**	2.72±0.01	2.40±0.10	2.46±0.04
Concrete	7.33±0.25	4.46±0.16	4.67±0.09	**4.16±0.07**	3.13±0.03	3.51±0.00	**2.89±0.02**	**2.89±0.01**
Energy	0.84±0.03	**0.46±0.02**	1.13±0.05	0.69±0.01	1.32±0.03	2.30±0.00	1.89±0.01	**1.04±0.01**
Yacht	1.58±0.37	0.57±0.05	1.08±0.06	**0.47±0.05**	1.39±0.14	1.75±0.00	**1.08±0.06**	1.14±0.09

4.4 Summary

In this chapter, a probabilistic neural network is developed based on the adaptive Gaussian mixture scheme in Chapter 2. The network's inputs, parameters, intermediate-layer neuron states, and outputs are all characterized with Gaussian mixture distributions. The Gaussian mixture states are refined adaptively before propagated through each layer, as triggered by two proposed criteria, to not only minimize the nonlinear distortion at the current layer but also ensure the fidelity of linear transformation at the next layer. Thereby, the predictive distributions of output can be inferred analytically without sampling or integration, to provide high-fidelity probabilistic uncertainty quantification. The gradients of the negative log-likelihood loss function with respect to all the Gaussian mixture parameters have also been derived analytically so that the proposed GM-PNN can be trained with any gradient-descent method. The GM-PNN in this work exhibits a state-of-the-art performance when benchmarked against other latest methods on a series of famous public datasets. Based on the success on benchmark datasets, next, the GM-PNN will be applied to address practical data-driven problems subject to uncertainties, as will be presented in the next chapter.

5. APPLICATION OF PROBABILISTIC NEURAL NETWORKS TO CONDITION MONITORING OF MANUFACTURING PROCESSES

This chapter describes the applications of the probabilistic neural network in Chapter 4 on two manufacturing process monitoring schemes. First, a robust tool wear monitoring scheme for turning processes is developed. Signal features that can best predict tool wear are extracted from sensors on the machine tool and selected using a systematic scheme. Two tool wear models are trained, one using a type-2 fuzzy network for interval uncertainty quantification and the other using the network in Chapter 4 for probabilistic uncertainty quantification. Secondly, a porosity monitoring scheme for laser AM processes is developed. A high-speed camera is used for in-process melt-pool sensing and CNN models are trained to directly learn melt-pool features for porosity prediction. The classical CNN models without uncertainty quantification are compared with the CNN models in which the probabilistic network is incorporated as an uncertainty quantification module. For both monitoring schemes, experimental results show that the probabilistic network not only achieves higher prediction accuracies of process conditions than the classical models but also provides more effective uncertainty quantification to facilitate the process-level decision-making in the manufacturing environment.

5.1 Tool Wear Monitoring of Turning Processes with Consideration of Uncertainties

The tool wear monitoring scheme is developed in this section. The sensor instrumentation setup, experiment design, signal processing and feature selection methods are detailed. The GM-PNN developed in Chapter 4 is applied to tool wear prediction based on the selected features and compared with a classical type-2 fuzzy neural network.

5.1.1 Instrumentation, Experiments Design and Feature Extraction

5.1.1.1 Instrumentation Setup

The tool wear testbed consists of a 20 HP Jones and Lampson CNC turret lathe with five sensors used for data collection during the turning operation. A diagram of the instrumentation setup is shown in Figure 5-1. A Hall-effect power meter is installed on the spindle motor drive.

Two PCB 307A quartz accelerometers are mounted on the tool holder in the Y-(cutting) and Z-(feed) directions. Additionally, two AE sensors with different frequency ranges are attached to the tool holder. All sensors used in this experimental setup are summarized in Table 5-1. The accelerometer and power meter signals are sampled by a National Instrument DAQ board while the AE signals are sampled by a Physical Acoustics PCI-2 board. All the five sensors adopted are affordable and non-intrusive to the machining process.

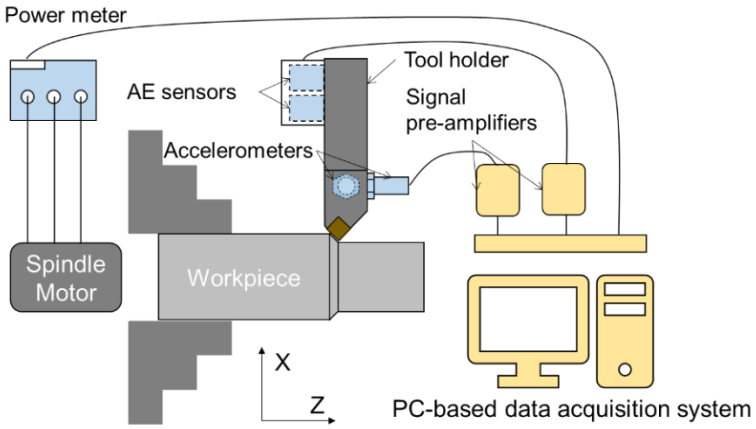


Figure 5-1 Instrumentation diagram of the test bed

Table 5-1 Information of the sensors used

Sensor	*Model Number*	*Specifications*	*Symbol*	*Sampling rate (Hz)*	*Purpose*
Power meter	Load Controls UPC	Response time: 0.5s	Power	10k	Measure the power input to the spindle motor
Accelerometer	PCB 307A	Freq. Range: 1-5000Hz Ampl. Range: ±50g Sensitivity: 100mV/g	AccY	10k	Measure the tool vibration in cutting direction
Accelerometer	PCB 307A	Freq. Range: 1-5000Hz Ampl. Range: ±50g Sensitivity: 100mV/g	AccZ	10k	Measure the tool vibration in feed direction
AE sensor	Physical Acoustics R15a	Freq. Range: 50-200kHz	AEL	2M	Measure the acoustic emission in low frequency range
AE sensor	Physical Acoustics WSa	Freq. Range: 100-900kHz	AEH	2M	Measure the acoustic emission in high frequency range

5.1.1.2 Experiment Design

To examine the generalization capability of the tool wear monitoring scheme that would be developed, three types of workpiece materials were used for tool wear tests: hardened 4140 steel (HRC35), Inconel 718 and Ti-6Al-4V. The 4140 steel and Inconel 718 workpieces were round bars of 6 inches in length and 1.5 inches in diameter, while the Ti-6Al-4V workpieces were round bars of 6 inches in length and 2 inches in diameter. The 4140 steel and Ti-6Al-4V were machined with Kennametal K68 carbide inserts (SPGN 422), while Inconel 718 was machined with Greenleaf WG-300 ceramic inserts (SNGN 452). Coolant was not used during any of these tests. The materials and cutting conditions of tool wear tests are listed in Table 5-2.

Table 5-2 Cutting conditions in tool wear tests

(Flank: tests used for the flank wear model; Crater: tests used for the crater wear model)

	Material	*Cutting Speed (m/min)*	*Feedrate (mm/rev)*	*Depth of Cut (mm)*	*Usage*	*Model*
Test #1	4140 Steel	90	0.20	0.25	Training	Flank
Test #2	4140 Steel	90	0.10	0.25	Training	Flank
Test #3	4140 Steel	90	0.20	0.15	Training	Flank
Test #4	4140 Steel	120	0.20	0.25	Training	Flank
Test #5	Inconel 718	120	0.075	0.25	Validation	Flank
Test #6	Inconel 718	120	0.075	0.15	Validation	Flank
Test #7	Inconel 718	100	0.075	0.25	Validation	Flank
Test #8	Inconel 718	120	0.10	0.25	Validation	Flank
Test #9	Ti-6Al-4V	100	0.075	0.75	Training	Crater
Test #10	Ti-6Al-4V	125	0.075	0.75	Training	Crater
Test #11	Ti-6Al-4V	100	0.10	0.75	Validation	Crater

Tool wear is measured after each tool pass. It is well documented that the machining of steel and Inconel is dominated by flank wear [225], while the machining of Ti-6Al-4V is dominated by crater wear [226]. The flank wear (*VB*) is measured by a Zeiss optical microscope while the crater depth (*KT*) is measured by a BRUKER 3D optical profiler. Each measurement is repeated five times, with the mean, minimum and maximum measurement values being used as the nominal tool wear and the uncertainty bounds, respectively. The inserts are replaced when flank wear reaches 200 μm or when tool chipping is observed, whichever occurs first.

In addition to the tool wear tests, since the sensor signals' dependence on cutting conditions need to be studied to establish a feature normalization scheme and the parameters in this scheme

also need to be determined experimentally, a set of normalization tests were also performed, as summarized in Table 5-3. The cutting conditions in Test#1, Test#5 and Test#9 are selected as the nominal conditions for the three materials. By machining at these conditions with fresh tools, the value of any feature can be extracted. By varying the cutting conditions around the nominal conditions, the signal feature's dependence on cutting conditions can be identified quantitatively, as will be discussed later.

Table 5-3 Cutting conditions for normalization

Material	*Cutting Speed (m/min)*	*Feedrate (mm/rev)*	*Depth of Cut (mm)*
4140 Steel	80, 90, 100	0.100, 0.150, 0.200, 0.250	0.15, 0.25, 0.35
Inconel 718	80, 100, 120	0.050, 0.075, 0.100, 0.125	0.15, 0.25, 0.35
Ti-6Al-4V	75, 100, 125	0.050, 0.075, 0.100, 0.125	0.25, 0.50, 0.75, 1.00

5.1.1.3 Feature Extraction

To represent the characteristic information embedded in the raw signal data in a compact and interpretable form, signal features are extracted from diverse signal processing schemes, including time-domain, frequency-domain and time-frequency-domain analyses. In this work, the power spectral density (PSD) is used for frequency-domain feature extraction. A four-level wavelet packet decomposition (WPD) with 6*th* order Symlets wavelet is utilized for time-frequency-domain feature extraction. The wavelet coefficients at each decomposition node can be viewed as a signal in the corresponding frequency band, from which wavelet features can be extracted. The list of features that are potentially useful for tool wear monitoring can be obtained from prior knowledge and literature review [134]. Among the sensors used, the power meter outputs an averaged power signal with minimal bandwidth, and thus only time-domain features are extracted from its signal. Since the AE signals are sampled at one or two megahertz and any complicated signal processing schemes should be avoided in consideration of computational efficiency, only time-domain and basic frequency-domain features are extracted. The vibration signals measured by the accelerometers are sampled at the kilohertz range, and are more suitable for feature extraction with different signal processing schemes. The candidate features are listed below by physical sensor:

Power meter: Time-domain features include the mean, RMS, standard deviation, skewness, kurtosis, peak-to-peak amplitude and crest factor.

Accelerometers: Time-domain features are the same as those of the power meter. Frequency-domain features include PSD mean (mean power level of the spectrum), frequency centroid (weighted frequency center of the spectrum), normalized PSD moment of inertia (spectrum distribution around the center) and normalized PSD entropy (flatness of the spectrum). Wavelet features include the powers of wavelet coefficients at different decomposition nodes.

AE sensors: Time-domain features are the same as those of the power meter. Frequency-domain features calculated from periodograms include the frequency centroid, peak spectrum amplitude and peak frequency.

5.1.2 Methodologies

The collected sensor signal data need to be processed properly before predicting tool wear. Firstly, the analog signals are preprocessed to obtain digital signals. Next, interpretable features are extracted from the digital signals in the time-domain, or transformations in other domains, as described in Section 5.1.1.3. The extracted features are normalized to minimize their sensitivity to variant cutting conditions, at which point, an optimal subset of features highly correlated to tool wear is selected. Finally, a predictive tool wear model is trained with the selected features. The architecture and data flow of the proposed monitoring system is summarized in Figure 5-2.

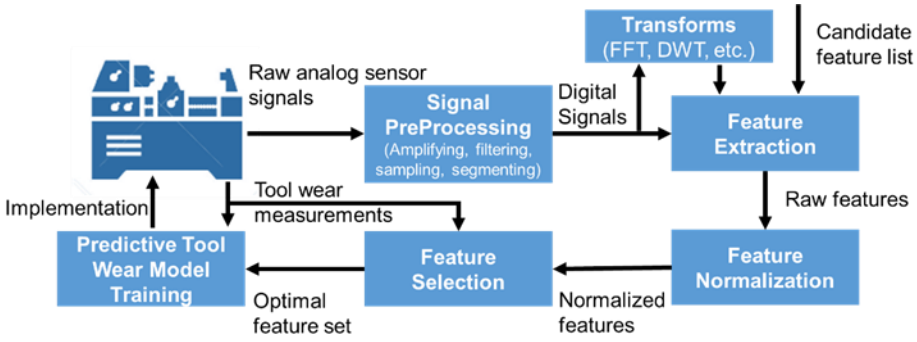


Figure 5-2 Overall architecture and data flow of the proposed tool wear monitoring system

5.1.2.1 Feature Normalization

Feature normalization is the key to making the monitoring system applicable to generalized conditions. The magnitudes of signal features vary due to one or more of the following factors: 1) the nature and units of different signals and features, 2) the changes of tools and workpiece

materials, 3) the changes of cutting conditions, and/or 4) the changes of process conditions such as tool wear. The idea of feature normalization is to properly scale the signal features into more uniform ranges to minimize the variations caused by the first three factors while preserving variations caused by the fourth factor. A normalization scheme is proposed as follows:

Assume that the raw signal feature at cutting speed V, feedrate f and depth of cut a_p is sf and the feature value at a nominal cutting condition with fresh cutting tool is known as sf_0, then the normalized feature value sf_N can be obtained as Eq (5.1).

$$sf_N = \frac{sf\left(V, f, a_p\right)}{sf_0 * \left(\frac{V}{V_0}\right)^{\alpha} \left(\frac{f}{f_0}\right)^{\beta} \left(\frac{a_p}{a_{p,0}}\right)^{\gamma}} \tag{5.1}$$

where α, β, γ are the coefficients to be identified for each feature and V_0, f_0, $a_{p,0}$ are the nominal cutting speed, feedrate and depth of cut at which sf_0 is extracted. The three ratio terms in the denominator are used to eliminate the influence of cutting conditions, while sf_0 accounts for the effects of raw feature magnitudes, machine and measurement setups, workpiece materials and cutting tools. It is expected that the normalized features of fresh tools are always close to 1, regardless of cutting conditions. Then, as tool wear increases, the normalized features deviate from 1 and give indicative information of tool wear.

For a given set of cutting tests using fresh tools at different cutting conditions, raw features can be extracted for each condition, and the coefficients (α, β, γ) can be determined from the following optimization in Eq (5.2).

$$\begin{aligned} &\textbf{find } \alpha, \beta, \gamma \\ &\textbf{to minimize } \sqrt{\frac{1}{n}\sum_{n}\left[sf_N\left(\alpha, \beta, \gamma\right) - 1\right]^2} \end{aligned} \tag{5.2}$$

where n is the total number of feature samples obtained from all the cutting conditions. The initial searching ranges of α, β and γ are suggested as $[-1,1]$, but can be broadened if necessary. After the coefficients α, β, γ are identified, they can be stored in a database. For a new workpiece material or cutting tool, only one cutting test with a fresh tool under the nominal cutting condition is required to calibrate sf_0.

5.1.2.2 Feature Selection

Even in a sensor fusion scheme, feeding a redundant set of features into a machine learning model might not improve, or could even degrade the model's performance. Instead, a compact feature set that best predicts tool wear should be selected. In addition, the signal preprocessing parameters (sampling rate and filter frequencies) do affect the features' correlation to tool wear and thus need to be determined together with the feature selection. In this work, the features and their preprocessing parameters are selected systematically in the following steps:

First, a preliminary feature selection is carried out to downselect candidate features. In this step, different signal preprocessing parameters are applied to the same feature and the optimal preprocessing parameters are selected as those maximizing the feature's correlation with tool wear. Features showing low correlation, even with their best preprocessing parameters, will be eliminated from the candidate list. Since the commonly used Pearson correlation coefficient can only assess the linear correlation [132], in this work, a feature's correlation with tool wear is instead measured by the coefficient of determination (R^2) of the best single-feature regression model among linear, parabolic or exponential functions. The preliminary selection procedure is presented in Figure 5-3. Since the training of tedious neural network models is not involved, this procedure, though exhaustive, can be executed efficiently and is scalable to large candidate pools of features and signal preprocessing parameters.

Next, a systematic frequency band analysis is performed. If a feature's correlation to tool wear depends on frequency bands, the above procedure in Figure 5-3 only gives the single best passband for that feature. The effect of frequency bands can be more systematically analyzed with the following three manipulations:

1) Merging frequency bands. If a feature has good correlations in several frequency bands, merging these separate bands into a larger one can give a more compact feature set and might enhance the feature's performance. This can be addressed by comparing the regression using the feature in the merged band with the regressions using the features in the original bands. The *F*-statistic and *p*-value from an analysis of variance (ANOVA) test can be used to assess regression models. The *F*-statistic indicates the statistical significance of the model while the *p*-value is the probability that the null hypothesis (the regression model is equal to a constant model) is true. If a larger *F*-statistic and a lower *p*-value are observed for the regression model with merged bands, it can be concluded that merging the frequency bands could be beneficial.

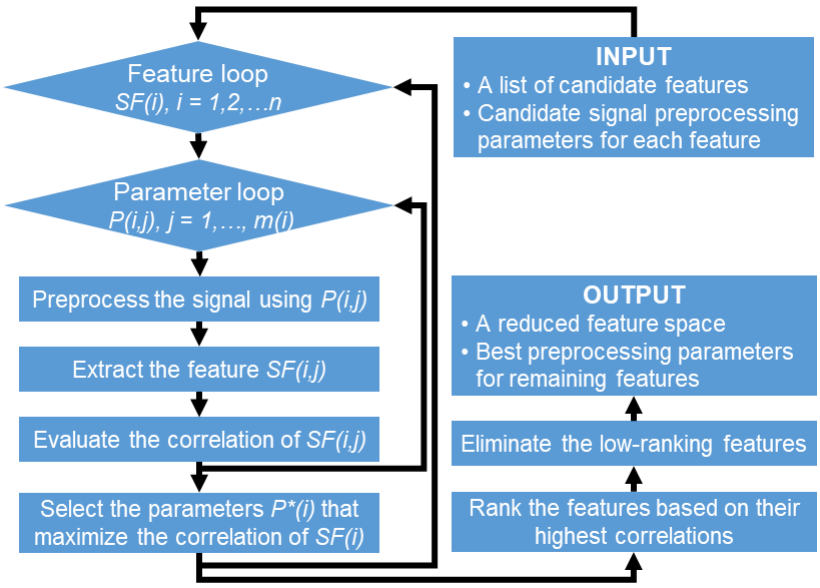


Figure 5-3 Flow chart of the coupled feature and preprocessing parameter selection

2) Multi-band analysis. Features at different frequency bands might have information complementary to each other. Instead of merging the bands, another strategy is to use the same type of feature in multiple bands to predict tool wear. This can be addressed by assessing the multi-band regression in Eq. (5.3):

$$y = a_0 + \sum_{i=1}^{n} a_i x_i \quad \begin{cases} x_i = \log\left(\text{Feature @ band \#i}\right) \\ y \text{ is the measured tool wear} \end{cases} \tag{5.3}$$

If a multi-band regression from Eq. (5.3) (n>1) is better than the single-band regression (n=1), then it can be said that using the features in multiple bands is beneficial.

3) Band feature fusion. Combining different types of features at the same frequency band may better describe the system behavior at that band, which can be assessed by a feature fusion analysis. Assume that three features are selected for the analysis, a permutation pool can be created via multiplication and/or division of the features with each other, as shown in Eq. (5.4):

$$\begin{aligned} &x_i = \log\left(\text{Feature \#i @ the band}\right) \quad i = 1,2,3 \\ &V = \left\{x_i, x_i^2, x_i x_j, x_i / x_j, x_i x_j x_k, x_i x_j / x_k\right\} \quad i, j, k \in \{1,2,3\}, \ \ i \neq j \neq k \end{aligned} \tag{5.4}$$

With three features, the size of the permutation pool is 19. Linear regressions by choosing items from the permutation pool are assessed as shown in Eq. (5.5):

$$y = a_0 + \sum_{i=1}^{n} a_i v_i \quad \begin{cases} v_i \in V \\ y \text{ is the measured tool wear} \end{cases} \tag{5.5}$$

A potential fusion of permutation terms is identified if R^2 of the regression is higher than those of single-term regressions. ANOVA test is also applied to ensure each permutation term in the identified regression model is above a significance level ($p<0.01$). Since a compact feature set is desired, the n in Eq. (5.3) and Eq. (5.5), which controls the number of bands and features, should be small ($n \leq 3$).

At last, the optimal feature set is selected. After the first two steps of feature selection, the tool wear model hasn't been involved yet nor has its performance been optimized in terms of input variables. Assuming that N features have been selected after the first two steps, the optimal feature set is subsequently selected using a backward elimination method:

(1) Exclude one feature from the N candidates, use the remaining $N-1$ features to train a tool wear prediction model and record the MSE (mean square error) as the performance index.

(2) Repeat (1) for all the N features and remove the feature whose $N-1$ complementary set has the lowest MSE, because this feature yields the minimum performance reduction when excluded from the tool wear prediction model.

(3) Set $N=N-1$ and repeat (1) ~ (2) until the minimum number of features left.

The above procedure is applicable regardless of which machine learning model is adopted as the tool wear model. Since the initial number of features N is small, the procedure will not be unacceptably time-consuming although the data-driven model training is iteratively involved.

5.1.2.3 Type-2 Fuzzy Basis Function Network

There are always uncertainties in the machining process; therefore, even after the same length of machining time, the tool wear could be different in two wear tests with the same cutting condition. Even with the same amount of tool wear, it is impossible to replicate the same feature values. While the tool wear models in the literature are mostly trained to minimize the prediction errors, for more reliable decision-making regarding tool wear, it is also of interest to know the magnitude of uncertainty associated with the crisp tool wear prediction at any given moment. For this reason, the GM-PNN developed in Chapter 4 will be applied to the predictive modeling of tool wear with uncertainties. To provide a performance baseline of uncertainty quantification, tool

wear models will also be constructed using the interval type-2 fuzzy basis function network (T2FBFN) proposed in [17], and the GM-PNN models will be benchmarked against the T2FBFN models. The T2FBFN is briefly introduced below.

Type-2 fuzzy basis function network is a two-layer network constructed using a fuzzy inference system, with interval type-2 fuzzy sets at the output layer. A T2FBFN with *n* input variables and *J* fuzzy rules can be formulated to approximate a nonlinear function in an interval form, as shown in Eq (5.6):

$$\begin{aligned} \tilde{y} &= \left[y_l,\ y_u \right] = \left[\mathbf{p}\mathbf{w}_l,\ \ \mathbf{p}\mathbf{w}_u \right] = \left[\sum_{j=1}^{J} w_l^j p_j\left(\mathbf{x}\right),\ \ \sum_{j=1}^{J} w_u^j p_j\left(\mathbf{x}\right) \right] \\ p_j\left(\mathbf{x}\right) &= \frac{\prod_{i=1}^{n} \mu_i^j\left(x_i\right)}{\sum_{j=1}^{J}\left(\prod_{i=1}^{n} \mu_i^j\left(x_i\right)\right)} \end{aligned} \tag{5.6}$$

where $x_1, x_2, \ldots, x_n$ are the input variables, $p_j\left(\mathbf{x}\right)$ is the fuzzy basis function in the *j*-th fuzzy rule, $\mu_i^j\left(x_i\right)$ is the fuzzy membership function, $\left[y_l,\ y_u \right]$ are the lower and upper bounds of the output, and $\left[\mathbf{w}_l, \mathbf{w}_u \right]$ are the lower- and upper-bound weighting factors in the sense of neural network, or the end points of interval type-2 fuzzy sets in the sense of fuzzy logic.

To build a T2FBFN, a training set of *N* samples $\left\{ \mathbf{x}\left(k\right), y_n\left(k\right), y_l\left(k\right), y_u\left(k\right) \right\}$ with *k=1,2, ..., N*, can be obtained from experiments, in which each vector $\mathbf{x}\left(k\right)$ contains *n* processed signal features, and $y_n(k)$, $y_l(k)$, $y_u(k)$ are the nominal value, lower-bound and upper-bound of tool wear measurements, respectively. Fuzzy rules are added to the network sequentially by searching fuzzy membership function parameters using generic algorithm (GA) to minimize the prediction error with respect to nominal tool wear measurements. The weighting factors $\left[\mathbf{w}_l, \mathbf{w}_u \right]$ are then determined from the constrained optimization problems in Eq. (5.7):

$$\begin{cases} \min\limits_{\mathbf{w}_l} \left\| \mathbf{P}\mathbf{w}_l - \mathbf{y}_n \right\|, & \mathbf{P}\mathbf{w}_l \le \mathbf{y}_l \\ \min\limits_{\mathbf{w}_u} \left\| \mathbf{P}\mathbf{w}_u - \mathbf{y}_n \right\|, & \mathbf{P}\mathbf{w}_u \ge \mathbf{y}_u \end{cases} \tag{5.7}$$

where the response matrix $\mathbf{P}\left(k\right) \in \mathbb{R}^{N \times J}$ is defined as $\mathbf{P}\left(k, j\right) = p_j\left(\mathbf{x}\left(k\right)\right)$ with *k=1,2, ..., N* and *j= 1,2, ..., J*, and *J* is the number of fuzzy rules that have been added where $\mathbf{y}_l$ and $\mathbf{y}_u$ are the vectors of lower- and upper-bound of tool wear measurements. The least square problems can be solved using the active-set method [17]. In brief, the T2FBFN captures the uncertainties in both the input-output (feature-wear) mapping and tool wear measurements by finding the weighting

factors using Eq. (5.7). The trained T2FBFN gives tool wear prediction in a lower- and upper-bound interval form enclosing the tool wear measurements, and the average of lower- and upper-bounds can be used as the nominal tool wear prediction.

Fluctuation in the model prediction is unavoidable, but monotonically increasing tool wear is expected in practice since the tool is never self-healing. To ensure monotonicity, the median value from the previous five tool wear predictions is compared to the present T2FBFN output, and the maximum is used as the corrected tool wear prediction. Once the corrected prediction reaches a certain value, it is latched so that the predictions afterward never drops below that value. After the correcting and latching mechanism, predictions of the T2FBFN can be assigned into discrete classes to indicate the quantized level of tool wear. The number of classes can be chosen based on the degree of uncertainties in the process and the visualization needs of users. The same post-processing will also be applied to predictions of GM-PNN.

5.1.3 Results and Discussions

5.1.3.1 Results of Feature Extraction, Normalization and Selection

After raw signal data were collected from the designed experiments and candidate features were extracted, the proposed feature selection procedure was applied. The features were selected by only analyzing the data of 4140 steel (Test#1~Test#4 in Table 5-2) and the effectiveness of selected feature set was validated for other materials.

First, the preliminary feature selection was applied to search the best signal preprocessing parameters for each candidate feature and eliminate the features with low correlations. The ranges of signal preprocessing parameters are provided in Table 5-4. To better capture the trend of spindle power, the raw signal was smoothed with a moving average filter before calculating the mean power feature. Remaining features from the power signal reflect the power fluctuation at different orders, and thus low-pass filters were used to remove noise while preserving the variation below the cut-off frequencies. The vibration and AE signals were processed with band-pass filters to inspect the features' correlation to tool wear at different frequency bands. For the vibration signals, the natural frequencies should be avoided as they are setup-dependent. In this work, the first natural frequency of the testbed was about 1000Hz, and thus only the bands below 1000Hz were considered. Since a node in the wavelet packet decomposition (WPD) inherently corresponds to a

frequency band, band-pass filters were not used for wavelet features.

Table 5-4 Admissible signal preprocessing parameters of candidate features

Sensor	*Features*	*Admissible Preprocessing Parameters*		
		Down-sampling rate (Hz)	*Filter Type*	*Filter Parameter*
Power	Mean	500, 200, 100, 50, 25	Moving average	Window size (s): 0.1, 0.2, 0.5, 1, 2, 5
Power	Others	500, 200, 100, 50, 25	Low-pass	Cut-off frequency (Hz): 5, 10, 25, 50, 100, 200
AccY & AccZ	All except WPD features	5000, 3333, 2500	Band-pass	Passband (Hz): 0-100, 100-200, …, 900-1000
AEL	All	500k, 333k	Band-pass	Passband (Hz): 50k-100k, 100k-150k, 150k-200k
AEH	All	1000k, 500k	Band-pass	Passband (Hz): 100k-150k, 150k-200k, …, 850k-900k

The features were normalized before feature selection. Figure 5-4 compares the mean spindle power feature before and after normalization. The normalization scheme collapsed the feature values from different cutting conditions into a uniform range, so that the feature's overall performance in different conditions could be evaluated through a single regression analysis. Figure 5-5 shows the correlation of normalized y-direction vibration RMS in different passbands. The RMS had a high correlation around the 300-700Hz frequency range, while the impact of the sampling rate was negligible. The observation that the sampling rate minimally impacts a feature's correlation to tool wear also holds for other features. Given this, it is preferred to use the lowest sampling rate compatible with the optimal filter parameters to minimize the burden of data transferring, processing, and storage during implementation.

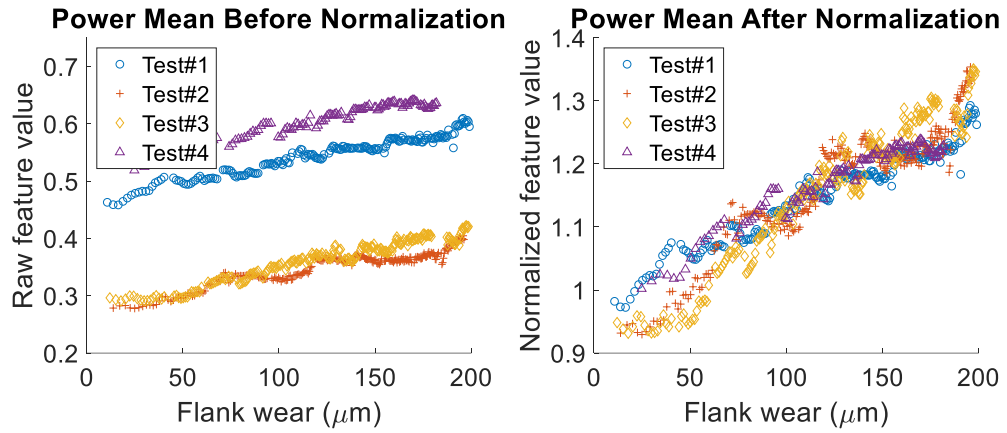


Figure 5-4 Comparison of the mean power feature before and after normalization

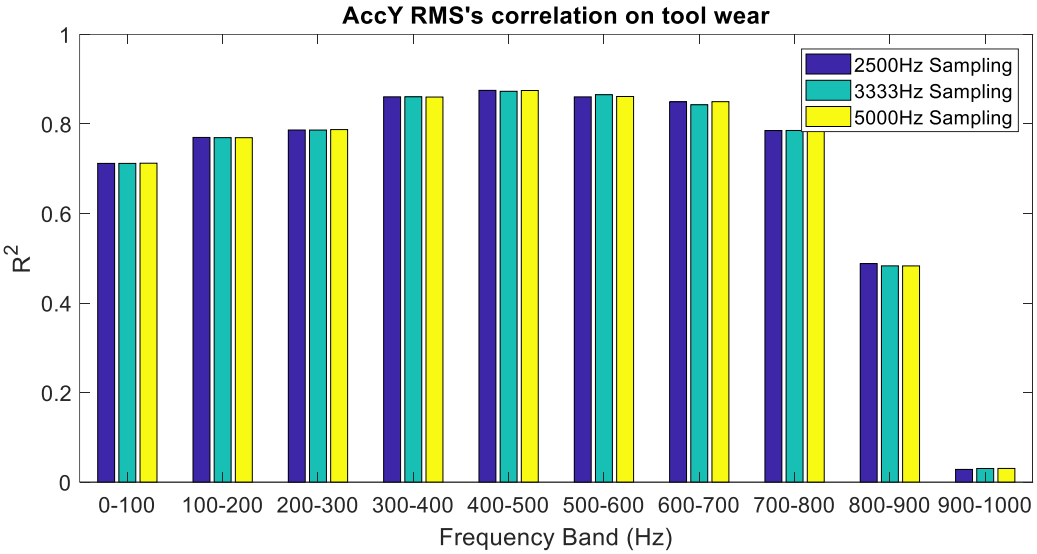


Figure 5-5 The correlation of y-direction vibration RMS against tool wear

In Table 5-5, the features with R^2 higher than 0.5 are listed. The power meter and the y-directional accelerometer (AccY) both had features highly correlated to tool wear (R^2>0.8). For vibration and AE signals, since the standard deviation (STD) was almost equal to RMS, and the peak-to-peak amplitude was more susceptible to noise, they could be eliminated if RMS had been considered. The z-directional accelerometer (AccZ) and the two AE sensors (AEL and AEH) only contributed features with moderate correlation (0.5<R^2<0.7). For industrial implementation, it is preferred to minimize the number of sensors to reduce the cost. Therefore, these sensors were excluded from further analysis in this work and only the features from the power meter and y-directional accelerometer (AccY) were selected.

Table 5-5 The features with R^2 higher than 0.5

Features	R^2	*Features*	R^2
Power mean: 2s moving average	0.89	*AccY wavelet coeff. power: node [3,2]*	0.86
AccY RMS/STD: 400-500Hz	0.88	*AccZ RMS/STD: 500-600Hz*	0.60
AccY peak-to-peak: 400-500Hz	0.80	*AccZ PSD mean: 500-600Hz*	0.57
AccY PSD mean: 600-700Hz	0.84	*AccZ wavelet coeff. power: node [3,2]*	0.60
AccY frequency centroid: 900-1000Hz	0.51	*AEL RMS/STD: 100k-150kHz*	0.65
AccY PSD moment of inertia: 900-1000Hz	0.50	*AEH frequency centroid: 200k-250kHz*	0.55

The AccY features (RMS, PSD mean, frequency centroid and PSD moment of inertia) were used for frequency band analysis. By merging frequency bands, the RMS achieved the highest *F*-stat in a 300Hz band, at 400-700Hz, while the PSD mean achieved the highest *F*-stat in a 200Hz band, at 500-700Hz. The frequency centroid and PSD moment of inertia were more statistically significant in wider bands, but their significances were not comparable to those of the RMS and PSD mean, as shown in Figure 5-6. These observations further confirmed that the AccY features' correlations to tool wear are band-dependent and it is necessary to perform the frequency band analysis. The RMS was used for multi-band analysis to see if combining the vibration amplitudes in several frequency bands could give better tool wear prediction. A similar multi-node analysis was applied to the wavelet coefficient power. As can be seen from Table 5-6, the regressions with three bands or nodes provided considerably better tool wear predictions, in terms of R^2 and root mean square error (RMSE), than those with single band or single node (n=1). Therefore, the features in the n=3 rows would be used as candidates for the optimal feature set selection.

Table 5-6 Multi-band analysis of y-direction vibration RMS and wavelet feature

	Best Features	R^2	*RMSE (μm)*
n=1	*RMS 400-700Hz*	0.89	16.29
n=2	*RMS 375-475Hz; RMS 500-700Hz*	0.91	14.46
n=3	*RMS 25-125Hz; RMS 375-475Hz; RMS 500-700Hz*	0.92	13.46
n=1	*WPD Node [3,2]*	0.86	16.86
n=2	*WPD Node [2,1]; WPD Node [4,1]*	0.89	15.85
n=3	*WPD Node [2,1]; WPD Node [4,1]; WPD Node [3,5]*	0.90	15.08

In addition, the three PSD features were used for the band feature fusion in Eq. (5.4) and (5.5) to see if the frequency centroid and moment of inertia could enhance the performance of PSD mean. The three PSD features created a total of 1159 combinations of regression for each

frequency band. Table 5-7 shows the results from PSD feature fusion. The first three rows show that a linear combination of the three PSD features in three frequency bands provided the highest correlation to tool wear with a *p*-value much less than 0.01. Since the R^2 achieved with feature fusion was considerably higher than that of PSD mean alone, it could be said that the feature fusion improves the performance of the PSD mean feature. Therefore, the three PSD features at the 75-775Hz band would be used as candidates for the optimal feature set selection.

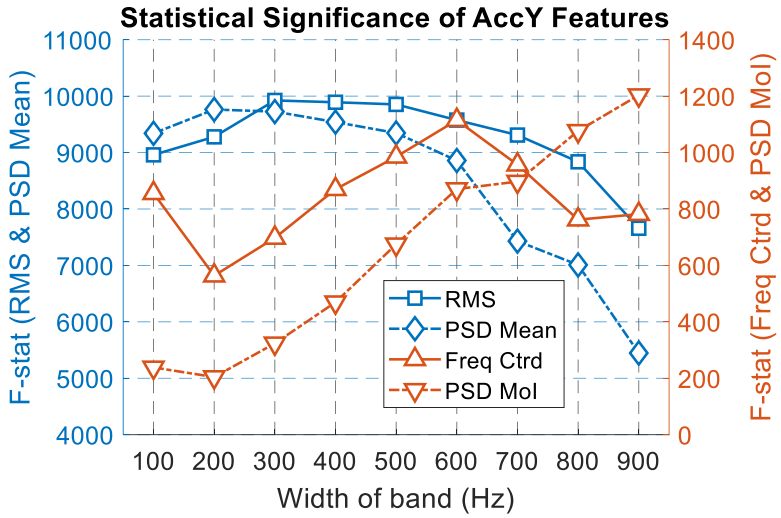


Figure 5-6 Statistical Significance of y-direction vibration features
(Freq Ctrd: Frequency centroid, PSD MoI: PSD moment of inertia)

Table 5-7 Results from frequency feature fusion (truncated to highest 5 entries)
x1: PSD mean, x2: frequency centroid, x3: PSD moment of inertia

Frequency Band	*Best Fit Model Form from Regression*	R^2	*p-Value*
75-775Hz	a0+a1*x1+a2*x2+a3*x3	0.920	3.67E-46
50-750Hz	a0+a1*x1+a2*x2+a3*x3	0.919	8.22E-55
0-800Hz	a0+a1*x1+a2*x2+a3*x3	0.913	4.46E-48
475-675Hz	a0+a1*x2+a2*x1^2+a3*x1*x2	0.908	1.05E-33
150-650Hz	a0+a1*x2+a2*x3+a3*x1^2	0.907	4.64E-51

Next, the backward feature elimination was applied to the 10 candidates (mean spindle power and 9 AccY features from the frequency band analysis) to determine the optimal feature set that could maximize the performance of tool wear prediction. The performance was evaluated by the

MSE between the tool wear measurement and the nominal prediction of T2FBFN. The T2FBFN model was trained using the 4140 steel test data and validated using the Inconel 718 test data. However, it was found that though the features extracted for Inconel 718 had similar levels of correlation, their slopes with respect to tool wear were slightly different from their 4140 steel counterparts, probably due to the variance of material properties. The difference of slopes can be compensated using Eq (5.8):

$$sf_{n,718} = \left(sf_{n,718} - 1\right) * k_{4140} / k_{718} + 1 \tag{5.8}$$

where $sf_{n,718}$ is the normalized feature of Inconel 718, k_{4140} is the baseline slope of 4140 steel feature, and k_{718} is the slope of Inconel 718 feature. This compensation makes the features from two materials have a consistent trend and hence can be used in the same tool wear model.

The result of feature elimination is given in Table 5-8. As can be seen, removing a few features reduced the RMSE, which indicated that a compact optimal feature set should be used for sensor fusion rather than using as many features as possible. The minimum RMSE was achieved with 6 features. However, from 7 to 4 features, the changes of RMSE were trivial. For simplicity, it was decided to use 4 features as the inputs of tool wear models. The 4 features were the power mean with 2-second moving average, the AccY RMS in 375-475Hz and 500-700Hz bands, and the frequency centroid in 75-775Hz band, which formed the optimal feature set.

Table 5-8 Result of backward feature elimination

Order of elimination	*Number of remaining features*	*Feature eliminated*	*RMSE after Elimination (μm)*
0	10	N/A	10.72
1	9	PSD Moment of inertia 75-775Hz	9.67
2	8	RMS 25-125Hz	8.35
3	7	Wavelet Coefficient Power Node [4,1]	7.86
4	6	PSD Mean 75-775Hz	7.70
5	5	Wavelet Coefficient Power Node [3,5]	7.71
6	4	Wavelet Coefficient Power Node [2,1]	7.81
7	3	Frequency Centroid 75-775Hz	9.02
8	2	RMS 375-475Hz	10.32
9	1	Power Mean	13.10
10	0	RMS 500-700Hz	N/A

5.1.3.2 Tool Wear Prediction and Uncertainty Quantification

With the above feature set, flank wear models were trained using the 4140 steel test data (815 samples) and validated using the Inconel 718 test data (376 samples). The GM-PNN newly developed in Chapter 4 and the T2FBFN were applied respectively to build tool wear models, so that the probabilistic uncertainty quantification capacity of GM-PNN could be compared with the interval uncertainty quantification capacity of T2FBFN. Although in Section 5.1.3.1, the optimal feature set was selected using T2FBFN, it was assumed that this feature set was also the optimal set for GM-PNN. When training the T2FBFN model, hidden nodes were added to the network sequentially until the MSE of validation dataset starts to increase (overfitting), and the outcome model have 60 hidden nodes. The GM-PNN model was then also trained with the same structure as T2FBFN (a single hidden layer with 60 hidden nodes), so that their performances could be compared at the same level of model complexity. Besides, same as in Section 4.3, the GM-PNN training was carried out using SGDM algorithm with a learning rate of 0.01 and a momentum of 0.9. The splitting thresholds for both D_{KL} and D_{CP} were selected as 0.01, and the number of components in the PDFs of $\mathbf{W}_l$ and $\mathbf{b}_l$ was set to be 2 for all layers. Since the cutting tool is never self-healing, the latching mechanism described at the end of Section 5.1.2.3 was applied to the outputs of both T2FBFN and GM-PNN models so that once the predicted tool wear reached a certain level, it would never drop below that level afterward.

The tool wear prediction performances of T2FBFN and GM-PNN models are summarized and compared in Table 5-9, and their predictions for the validation set (Inconel 718 tests) are compared in Figure 5-7. As can be seen, the T2FBFN model achieved a decent performance. Its nominal predictions tracked the tool wear measurements closely with a R^2 of 0.97. Besides, the lower and upper bounds of T2FBFN predictions always enclosed the tool wear measurements, which means the prediction intervals were guaranteed to contain the true tool wear values. As will be shown later, this enables reliable decision-making about tool change. On the other hand, the performance of the GM-PNN model was even better, with a considerably higher R^2, smaller RMSE and lower negative log-likelihood (NLL). Note that the likelihood of T2FBFN was estimated by assuming a uniform distribution within its prediction intervals. In addition, the 95% and 99% confidence intervals (CI) were also extracted for GM-PNN from its predicted PDFs. As shown in Figure 5-7, the CIs of GM-PNN were significantly narrower than the intervals of T2FBFN, while

the true tool wear values were still consistently enveloped. Therefore, the GM-PNN quantified the uncertainty in tool wear less conservatively without compromising reliability.

Table 5-9 Comparison of tool wear monitoring model performance

	Validation R^2	*Validation RMSE*	*Avg. NLL*	*Avg. Interval Width*	*Tool change Margin*
T2FBFN	0.974	6.69 µm	3.24	27.9 µm	6.2%
GM-PNN	0.990	3.77 µm	2.66	95%: 13.6µm 99%: 18.0µm	95%: 4.2% 99%: 5.3%

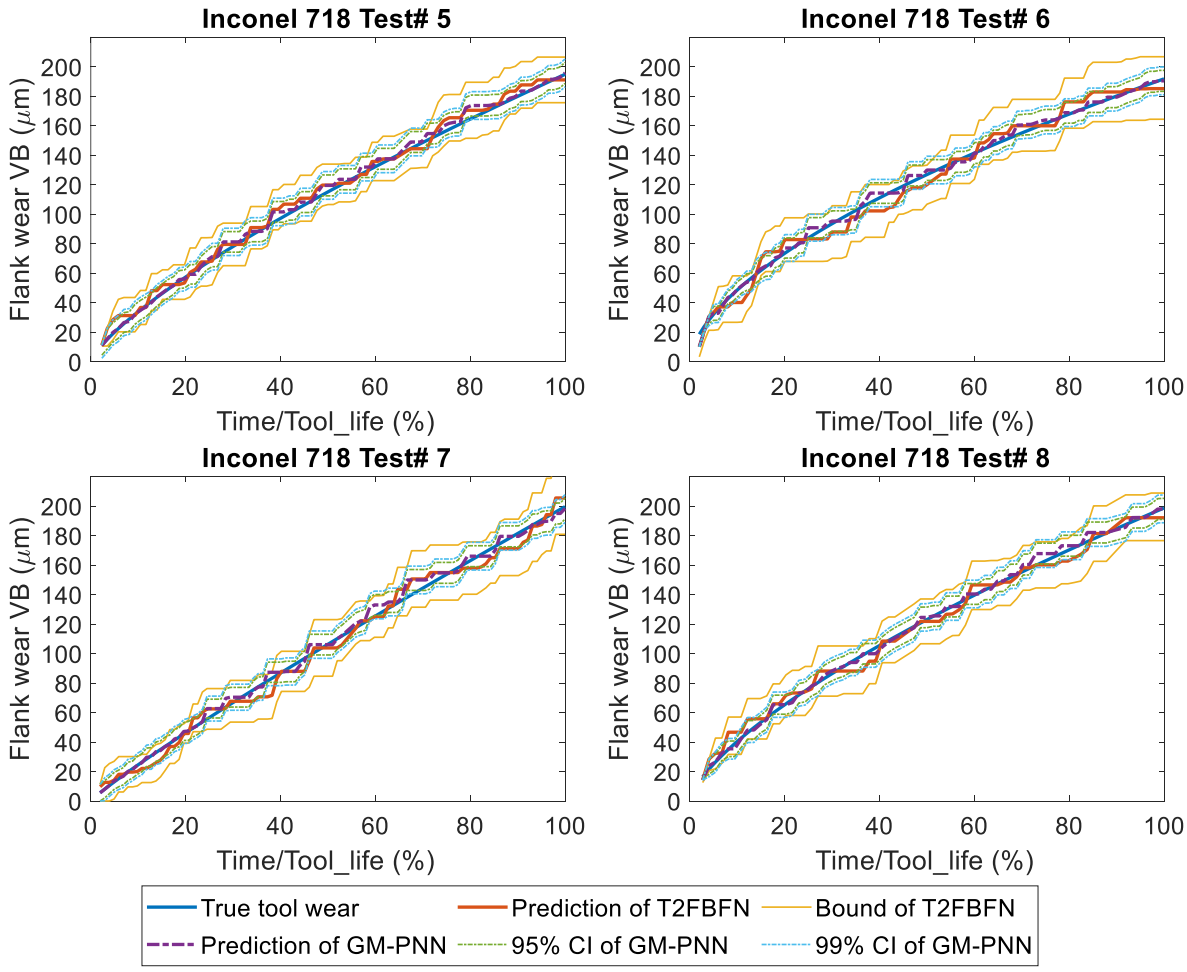


Figure 5-7 Tool wear prediction for the selected tool wear tests

While both of GM-PNN and T2FBFN could provide uncertainty quantification around their nominal tool wear predictions, for industrial users, it is of more interest to know when to change

tools than knowing the exact values of tool wear. To address this concern, for either model, once the upper-bound prediction exceeded the tool life limit, which is 200 μm in this work, a tool-change alert would be issued. The percentage by which the true tool wear was below the tool life limit when an alert issued was defined as the tool change margin, as Eq. (5.9):

$$\text{margin} = \frac{\text{Tool life limit} \ - \ \text{tool wear measurement when alert issued}}{\text{Tool life limit}} \times 100\% \qquad (5.9)$$

The tool change margins of T2FBFN are shown in Figure 5-8. In average, the T2FBFN model issued the alert 12.4 μm ahead, which is 6.2% of the tool life limit. By considering uncertainty in the T2FBFN model, the actual tool wear was always lower than its upper-bound prediction and thus would never exceed the tool life limit when the alter was issued. For tool wear monitoring, false negative is more undesirable than false positive. The upper-bound-based false-positive alert with acceptable lead is conservative, but could reliably prevent any premature tool failure. The tool change margins of GM-PNN based on its upper bound of CIs are shown in Figure 5-9. As can be seen from Figure 5-9 and Table 5-9, the tool change margin of GM-PNN was also smaller than that of T2FBFN, which means the monitoring scheme based on GM-PNN would better take advantage of the usable life of a tool while still effectively preventing tool failure.

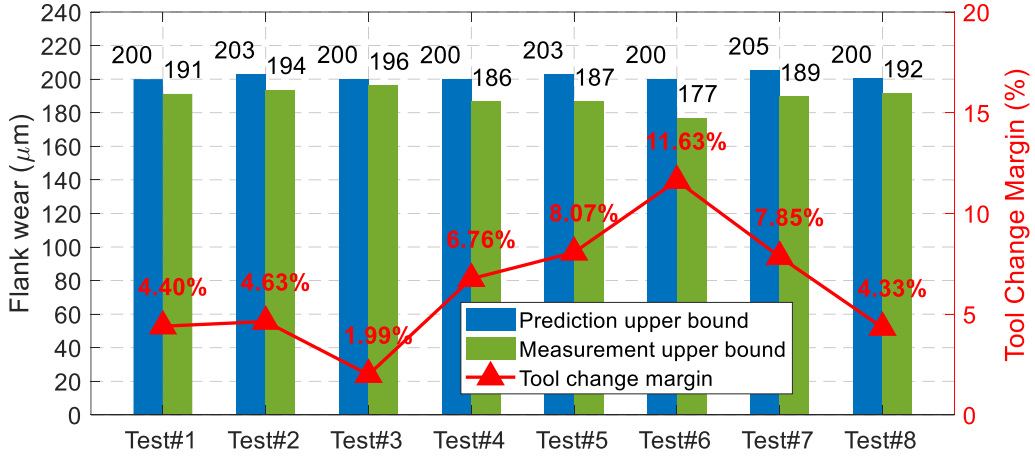


Figure 5-8 Comparison of the T2FBFN prediction upper bounds and tool wear measurements when the tool change alert issued

As for the machining of Ti-6Al-4V using a carbide tool, tool wear is dominated by crater wear rather than flank wear and hence the tool wear measurements and signal feature behaviors are different. Therefore, separate crater wear prediction models were trained with the same set of

features but using the Ti-6Al-4V test data (354 samples). For crater wear prediction, the GM-PNN model was also performing slightly better than the T2FBFN model. While both models had R^2 of 0.97, the prediction intervals of GM-PNN (0.67 μm for 95% CIs and 0.73 for 99% CIs) were narrower than the intervals of T2FBFN (0.79 μm). The improvement of GM-PNN models over T2FBFN models for both flank wear and crater wear is mainly because the T2FBFN uses a hard constraint to enforce its prediction intervals to enclose all the data. In contrast, GM-PNN addresses uncertainty from a probabilistic perspective to learn the model parameters that maximize the likelihood in describing the data, which is a soft constraint less prone to outliers.

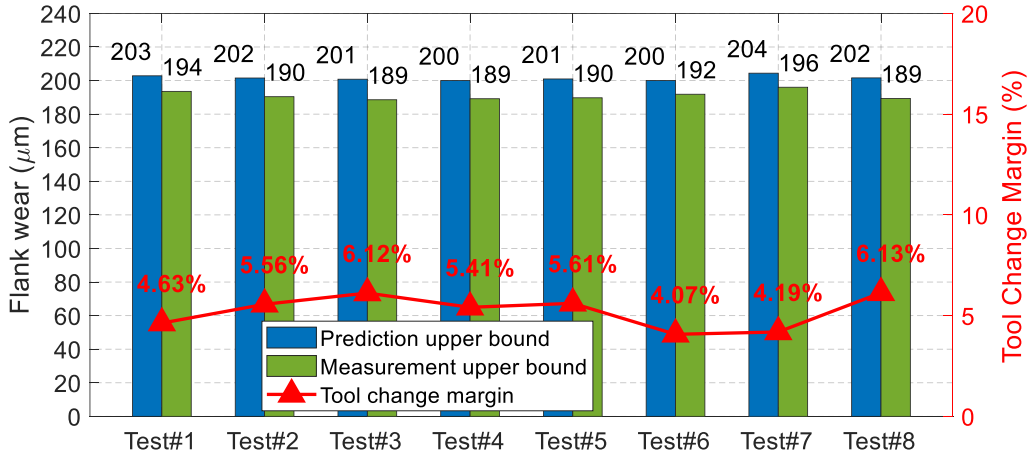


Figure 5-9 Comparison of the GM-PNN prediction upper bounds (99% CI) and tool wear measurements when the tool change alert issued

5.2 Porosity Monitoring of Laser Additive Manufacturing Processes

The porosity monitoring scheme is developed in this section. The melt-pool sensing setup, experiment design, porosity measurement and image processing methods are presented in detail. A compact CNN architecture is designed for porosity prediction. The CNNs with and without the uncertainty quantification module based on GM-PNN are compared.

5.2.1 Instrumentation and Experiments

5.2.1.1 Instrumentation Setup

The non-contact sensors that are most widely used for laser AM process monitoring include photodetector, infrared camera, and high-speed digital camera, because they are all sensitive to the process, and do not interfere with the AM processes. The high-speed digital camera is able to sense the 2-D melt-pool morphology and brightness and compared with other sensors, it offers the widest range of choices of specifications, optics (lenses and filters), interfaces, and software to configurate the monitoring scheme. Owing to its informativeness and flexibility, the high-speed digital camera is selected in this study for instrumentation of the laser AM system.

A digital camera is mounted coaxially to the process laser beam on an OPTOMEC LENS750 system (with 500W fiber laser), as shown in Figure 5-10. The camera used is a DMK 33UX174 monochrome industrial camera with a USB 3.0 interface, which can record 640×480-pixel images at 395 fps (frames per second). The resolution of camera images in this setup is calibrated as 124 pixel/mm and the field of view is about 5.2×3.9 mm. The main difficulty of laser AM process monitoring is that the coaxial-image contrast often exceeds the dynamic range of cameras, i.e., the melt pool is too bright and tends to saturate the sensor, while the surrounding is too dark to be sensed. To alleviate this issue, a narrow bandpass filter with the center wavelength of 532 nm is mounted in front of the camera to reduce the impact of melt-pool irradiations, especially those from the plasma, and a 200 mW green laser with 532 nm wavelength is used to illuminate the melt-pool surrounding.

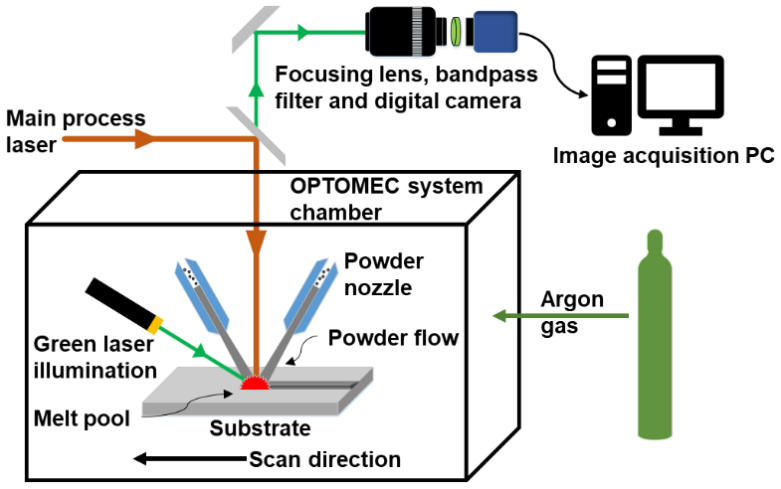


Figure 5-10 The instrumentation setup of direct laser deposition monitoring

5.2.1.2 Experiment Design

To study the porosity monitoring of the laser AM process, a set of single-layer-single-track direct laser deposition experiments were designed and conducted using sponge Titanium powders, as summarized in Table 5-10. The wavelength of the fiber laser was 1066 nm and the diameter of the laser spot was 660 μm. Coaxial melt-pool image data under various laser powers and laser scan speeds were collected by the camera from these designed experiments. Given the frame rate limit of the camera (395 fps), the laser scan speed was reduced to 1~4 mm/s (much lower than normal operations [227]) to ensure sufficient space resolution (2.5~10 μm) between image frames to capture small pores. All the deposition tracks were 15 mm long and the number of melt-pool image samples collected in each experiment depends on the duration, i.e., the track length divided by the scan speed.

Table 5-10 Experimental conditions in direct laser deposition

Test#	*Speed (mm/s)*	*Power (W)*	*Samples*	*Quality Inspection*	*Overall Volume Porosity*
1	1	250	5694	Cross sections	14.9%
2	2	250	2842	Cross sections	10.7%
3	4	250	1422	Cross sections	8.2%
4	4	150	1407	Cross sections	5.9%
5	2	250	2853	X-ray CT	12.2%

The image data collected by cameras need to be paired with the interior quality attributes of interest, such as the size and location of pores and volume porosity percentage. These attributes could be inspected either destructively (cross-sectioning and microscopy) or nondestructively (X-ray computed tomography). In the destructive approach, the AM specimen was cut along the laser scan direction to obtain longitudinal cross-sections, and then an optical microscope was used to observe the exposed pores on the cross-section. Since porosity is always scattered inside the specimen, two or three cross-sections were inspected for each specimen. However, because precise sectioning of the specimen is difficult and the number of cross-sections is limited, the destructive approach may omit the pores between obtained cross-sections and only provide an approximated distribution of porosity. In the nondestructive approach, the AM specimen was inspected using X-ray computed tomography (CT) to detect interior porosities. The X-ray CT can construct an accurate 3-D distribution of porosity, which makes it much more preferable than the destructive

approach. However, it is also much more expensive and time-consuming. In this study, four experimental samples were inspected using the cross-sectioning method to evaluate the volume porosity level, while one experimental sample was inspected using X-ray CT to get the accurate porosity measurement for porosity occurrence prediction.

5.2.2 Methodologies

5.2.2.1 Porosity Measurement

Before applying any supervised learning technique to build the porosity monitoring model, each coaxial melt-pool image collected by the high-speed camera needs to be paired with the porosity attributes of interests. To extract porosity information from the raw microscope images of cross-sections, a set of image processing tools has been developed in MATLAB. The procedure of image processing is discussed below and illustrated in Figure 5-11:

(1) The consecutive cross-section images are stitched using a SURF (speeded-up-robust-features)-based image registration method [229], given that a single microscope image cannot cover the length of a whole cross-section. In the SURF algorithm, the points of interest (the points differ in intensity to surrounding regions) on a microscope image are localized in the blob response map. The interest points with similar descriptor vectors on two consecutive images are detected as matching points and the location offset of the later image with respect to the prior image can be computed accordingly by the difference of coordinate of matching points. Then the two images can be automatically concatenated at pixel-level accuracy to minimize the error in locating porosity.

(2) The deposition track may have an incline angle on the stitched image if the specimen was not strictly horizontal when placed on the microscope stage. This inclination (if any) can be corrected by measuring the angle of the flat top surface of the substrate and rotating the image. The rotated image is then cropped from the beginning to the end and from the top to the bottom of the deposition track to exclude areas of the substrate and sample holder, which are not in the region of interest of the cross-section. After the rotating and cropping, the horizontal pixel coordinate on the image could be used as a measure of longitudinal distance from the start of a deposition track along the laser scan direction.

(3) The rotated and cropped image is binarized using the Otsu's threshold and the dark regions on the binary image are recognized as candidate porosities on the cross-section, except the dark background outside the cross-section. It is found that there are two types of dark regions observed on the cross-sections: 1) pores caused by entrapped gas during the process, which have circular or near-circular shapes and are the targets to be detected; 2) cracking and scratch marks introduced when sectioning the specimens, which usually have irregular shapes and should be excluded. Therefore, filter conditions are applied, so that only the candidate dark regions with an extent (ratio of pixels in the dark region to pixels in the total bounding box) larger than 0.3 and eccentricity (eccentricity of the ellipse that has the same second-moments as the region) less than 0.95 will be accepted as true gas porosity, while the irregular dark cracks, scratches, and textures on the cross-section will be automatically excluded, as illustrated in Figure 5-12.

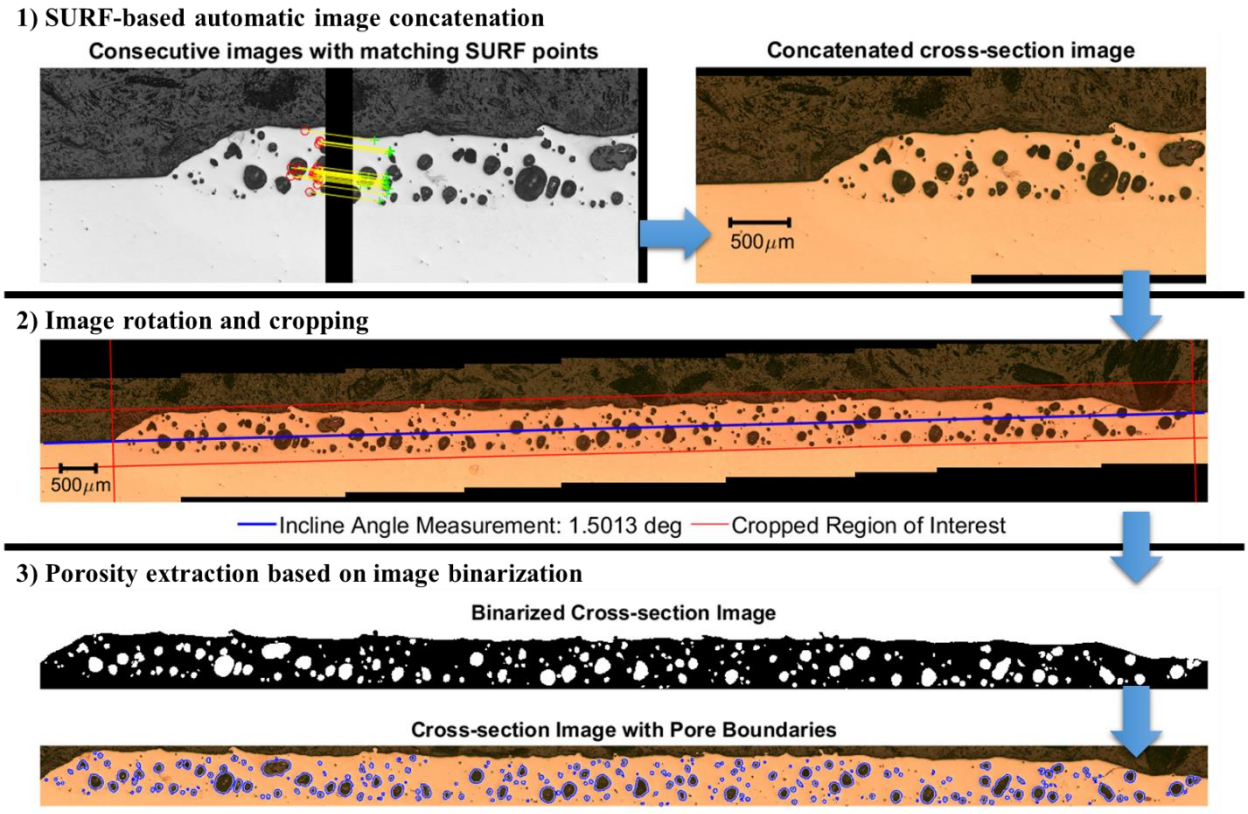


Figure 5-11 Image processing procedure for porosity extraction from cross-section images

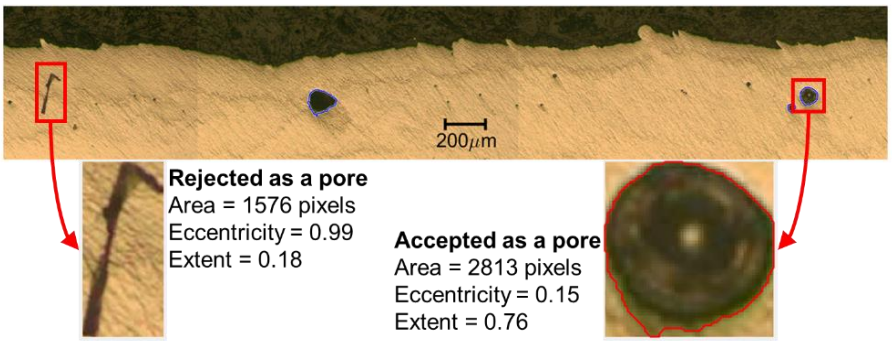


Figure 5-12 Illustration of the filtering to recognize true porosity

The X-ray CT data can be processed in a similar way, except that the concatenation of images and filtering out of irregular areas will not be necessary because the X-ray CT directly provides complete 3D data sets without cross-sectioning the sample. From the processed cross-section or CT data, the porosity attributes to be measured at each longitudinal position includes the porosity status (0: no pore, 1: pore), pore size (area or diameter), and volume porosity percentage. Each melt-pool image from the coaxial camera will be labeled with the measured attributes at the corresponding position, under the assumption that the image frame index can be linearly mapped to the longitudinal distance along the track. The labeled image forms an input (coaxial melt-pool image)-output (porosity attributes) data pair that can be used as a training sample for supervised machine learning. The porosity inspection and input-output data pairing are illustrated in Figure 5-13.

5.2.2.1 Convolutional Neural Network

Directly correlating melt-pool measurement data, such as melt-pool size, to porosity results in poor fidelity. Therefore, to map the melt-pool images to porosity occurring, the convolutional neural network (CNN), which has proven to be powerful in learning intricate features from high-dimensional data and is the core of deep learning, is adopted to learn spatial patterns from the input data (melt-pool image) to predict the output (porosity attributes).

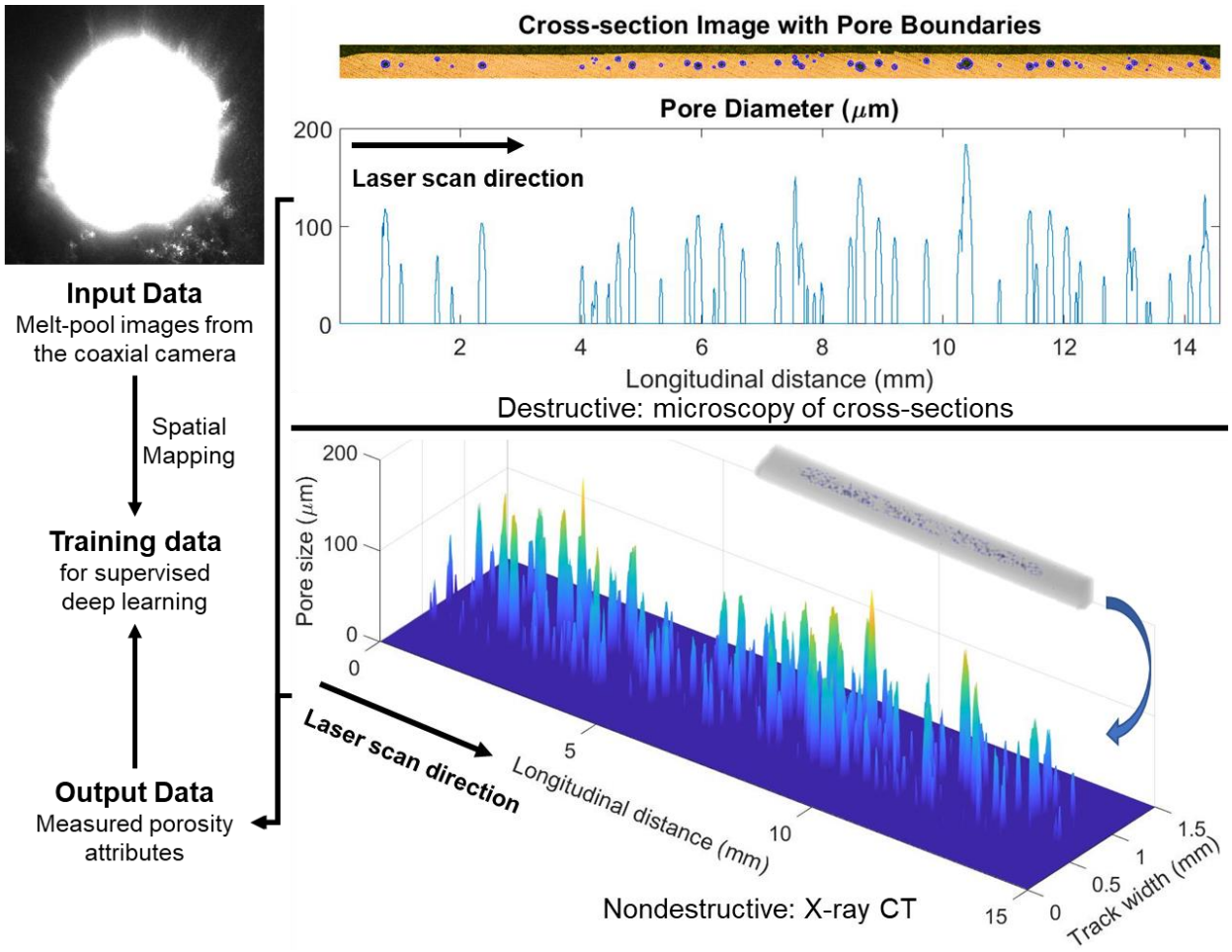


Figure 5-13 Illustration of quality inspection and data preparation

(Right top: the porosity information from a destructive cross-section; right bottom: the porosity information from a nondestructive X-ray CT; Left: the coaxial melt-pool image and pairing of input-output data)

To make this chapter more self-contained, the fundamentals of CNN are briefly discussed below. The grayscale images from the coaxial camera are cropped to 224×224 pixels with the melt pool in the center (see Figure 5-13), as input to the network. The input image is firstly processed by a set of filters convoluted with the input, from which the receptive field in the l-th convolutional layer and with the j-th filter can be represented as:

$$\mathbf{x}_j^l = \sum_{k=1}^{K_l} conv2\left(\mathbf{x}_k^{l-1}, \mathbf{w}_{l,j,k} \mid s_l\right) + b_{l,j} \qquad j = 1, \ldots, J_l \tag{5.10}$$

where $\mathbf{x}^{l-1}$ is the input from the previous layer (the original input image if l=1, or the output of the previous layer if l>1), $\mathbf{w}_{l,j}$ is the weight matrix of the j-th filter in the l-th layer, and $b_{l,j}$ is the bias

for the *j*-th filter. The $\mathbf{x}^{l-1}$ and $\mathbf{w}_{l,j}$ has the same depth K_l, which depends on the number of channels in the input image (3 for RGB images and 1 for grayscale images) if l=1 or the number of filters in the previous layer (i.e., $K_l = J_{l-1}$) if l>1. The *conv2* denotes the 2-dimensonal discrete convolution of $\mathbf{x}_k^{l-1}$ with filter $\mathbf{w}_{l,j,k}$ at depth slice k using a stride of s_l. Adding up the convolutions in all the K_l slices forms the receptive field for the *j*-th filter and stacking the receptive fields of all filters forms the feature map $\mathbf{x}^l$ with a depth of J_l. The feature map reflects the image's response, i.e., level of activation, to the features defined by the filters.

To address the issue that the distribution of feature map activations changes during training as the filter coefficients change, which is referred to as internal covariate shift, the activations in each slice of feature maps are scaled via a batch normalization operation as [230]:

$$\hat{x} = \frac{x - \mu_B}{\sqrt{\sigma_B + \varepsilon}} \qquad (5.11)$$
$$y = \gamma \hat{x} + \beta$$

where x is the activations before normalization, y is the normalized activations, μ_B and σ_B are the mean and variance of x over a training mini-batch, ε is a parameter that helps to improve numerical stability when σ_B is very small, and the offset β and scale factor γ are learnable parameters that allow for the possibility that activations with zero mean and unit variance are not optimal for the following layer. By scaling the activations into uniform ranges, higher learning rates can be used to accelerate the deep network training. And to introduce nonlinearity into the network, ReLU is often used as the activation function. Compared to sigmoid functions, ReLU alleviates the vanishing gradient issue for deep network training. In addition, a max-pooling layer is used after each convolutional layer, which substitutes the activations in a sub-region of feature map (defined by the pooling filter size) with the maximum value in that region. The pooling layer downsamples the feature map (if stride>*1*) and introduces translation invariance by making the feature activations less sensitive to their exact location. The feature map after pooling, which is a compressed abstract representation of the original image, is used as input to the next convolutional layer. The output of the last pooling layer will be rearranged as a feature vector and a set of fully-connected layers are cascaded to learn a mapping from feature to porosity.

The CNN is trained through a backpropagation algorithm with stochastic gradient searching

called Adam [231]. Let the learnable parameters (e.g., convolutional filter coefficients, fully connected layer weights) be $\boldsymbol{w}$ and the loss function to be minimized be L, then the learnable parameters are updated using the following rules:

$$\begin{aligned} w_t &= w_{t-1} - \alpha\, m_t \Big/ \left(1-\beta_1^t\right) \Big/ \left(\sqrt{v_t \Big/ \left(1-\beta_2^t\right)} + e\right) \\ m_t &= \beta_1 m_{t-1} + \left(1-\beta_1\right)\left\langle \nabla_w L\left(w_{t-1}\right)\right\rangle_B \\ v_t &= \beta_2 v_{t-1} + \left(1-\beta_2\right)\left\langle \nabla_w L\left(w_{t-1}\right)\right\rangle_B^2 \end{aligned} \tag{5.12}$$

where $\left\langle \nabla_w L\left(w_{t-1}\right)\right\rangle_B$ is the gradient of loss function over a training mini-batch, m_t and v_t are the first and second-moment variables and α, β_1, β_2, e are learning hyperparameters. In addition to the updating of learnable parameters, the performance of a CNN also depends on the selection of hyperparameters, including the number of layers, the size, stride, and number of filters in each convolutional layer, size and stride of pooling, number of neurons in fully connected layers, and the learning hyperparameters in training algorithm. Generally speaking, increasing the number of layers enables higher-level abstraction of data, increasing the number of filters allows the learning of more feature representations, and using larger strides yields faster data compression through the network. In this work, a compact CNN architecture is designed, in which some hyperparameters vital to the model performance are selected using cross-validation analysis, as will be shown in the next section.

5.2.3 Results and Discussions

5.2.3.1 Design of Deep Learning Model Structure

The direct laser deposition of sponge Titanium powders creates porous specimens, for which it is of interest to know what the volume porosity is and where the porosity occurs. The latter is more challenging as it requires datasets with accurate 3-D porosity measurement to train the predictive model, which means all the pores scattered inside a specimen need to be captured to minimize the mislabeling of data. Since in Table 5-10, only Test#5 inspected by X-ray CT meets this requirement, this dataset was used for the detection of porosity occurrence, while the other four cross-sectioning datasets were used for the prediction of volume porosity.

A compact CNN architecture was designed, as shown in Table 5-11, in which most of the structure hyperparameters are similar to those in the famous Alexnet [102], because its structure

has proved to be powerful in learning representations from images. However, as the Alexnet is aimed to classify images into 1000 categories while the porosity monitoring model is only used for binary classification of two categories (0: no pore, 1: pore), it should be possible to build the monitoring model with fewer features. Hence, the number of filters and fully connected neurons were reduced, which resulted in the CNN having significantly fewer learnable parameters (about 0.3 millions) than the Alexnet (about 61 millions) and thus it can be trained with much smaller data sets. Besides, batch normalization and ReLU function were used between the convolution and max pooling operations in each layer, and the hyperparameters in Eq (5.12) were adopted from the recommendations in [231], i.e. $\alpha=0.001$, $\beta_1=0.9$, $\beta_2=0.999$, $e=10^{-8}$.

Table 5-11 Architecture of the convolutional neural network for direct laser deposition

Layer	*Filter Size*	*Number of filters*	*Stride*	*Max Pooling*
Image Input	224×224 grayscale images			
Convolution	11	48	4	3×3, stride of 2
Convolution	5	64	2	3×3, stride of 2
Convolution	3	96	1	3×3, stride of 2
Convolution	3	96	1	3×3, stride of 2
Convolution	3	64	1	3×3, stride of 2
Fully Connected	64 output channels			
Fully Connected	32 output channels			
Fully Connected	2 output for classification			

To train the CNN model, each melt-pool image corresponding to a longitudinal position along the laser scan direction was paired with the class label obtained from porosity inspection. It is intuitive that all the melt-pool images with a pore should be labeled as true (true = pore, false = no pore). However, considering the dynamics of melt pool and the resolution limit of camera, only the pores above a certain size could produce a noticeable impact on the melt pool and thus become detectable through the camera. Hence, it might be necessary to set a threshold of pore size when preparing the labeled training samples. For example, if using a threshold of 15 μm, the pores smaller than 15 μm will be neglected (true: pores≥15 μm; false: no pore or pores<15 μm). This pore size threshold, as a hyperparameter of the model, was selected using a 10-fold cross-validation analysis. The 2842 data samples in Test#5 were randomly divided into 10 subsets of the same size. Given a threshold to determine the class labels of data samples, one subset out of the

ten was selected for validation each time and the rest nine subsets were used for training. The classification accuracy of the trained CNN model on the validation subset was recorded. This data partition was repeated ten times until each subset had been used for validation once, and the average classification accuracy of the 10 folds was used as the performance index of that threshold. The results of the cross-validation are given in Table 5-12. It can be seen that using a threshold of 10μm gives the best results. This might be because the pores smaller than 10 μm could not produce a detectable pattern on the melt pool images or because the CT resolution was 9.95 μm/pixel and thus those pores smaller than 10 μm have only 1 pixel in the tomography data, which was not a reliable measurement. Using a too large pore size threshold would also degrade the performance because some image samples with porosity patterns were mislabeled as false. Therefore, 10 μm was selected as the optimal pore size threshold.

Table 5-12 Cross-validation of the pore size threshold

Pore size threshold	*0 μm*	*10 μm*	*20 μm*	*30 μm*	*40 μm*
Mean classification accuracy	89.62%	91.06%	90.85%	89.48%	88.00%

The above analysis was based on the CNN structure of 5 convolutional layers, as in Table 5-11. The number of layers is also a vital hyperparameter that affects the network performance. Could making the network deeper (more layers) improve the performance, or could the network be further simplified by using fewer layers without degrading the performance? Another cross-validation analysis for the number of convolutional layers was conducted. The number of layers was reduced to 3 or 4 by removing the last one or two convolutional layers, or increased to 6 or 7 by cascading more convolutional layers with 3×3 filters after the 5th layer. The results of this cross-validation are given in Table 5-13. It can be seen that using too few layers (like 3 layers) didn't give good performance, but using too many layers (like 7 layers) would also slightly degrade the performance. Based on the cross-validation, the CNN with 5 convolutional layers and the pore size threshold of 10 μm was the best model for predicting the porosity occurrence.

Table 5-13 Cross-validation of the number of layers

Number of convolutional layers	*3*	*4*	*5*	*6*	*7*
Mean classification accuracy	87.23%	90.22%	91.16%	90.39%	90.39%

5.2.3.2 Results of Porosity Detection

The above CNN structure design was carried out using a classical CNN without considering uncertainty. This CNN would be used as a baseline model. In practice, an AM process does involve substantial uncertainties [232], such as the fluctuating laser power absorption, varying material properties in powders, and noise radiations from the environment that corrupt melt-pool images. Therefore, after the optimal CNN structure was determined, the GM-PNN developed in Chapter 4 was combined with CNN to create a porosity model with uncertainty quantification capacity. Assuming that the convolutional layers in the baseline CNN had been well-trained to extract representative features of the melt pool, a GM-PNN module was used to replace the fully connected layers (i.e., the last three rows in Table 5-11) without altering the convolutional layers. The GM-PNN module took the 96 features from the last convolutional layer as input to predict the porosity status. It had the same structure as the original fully connected module, while all the connection weights in it are parameterized as Gaussian mixtures. The GM-PNN was retrained using the features extracted by the baseline CNN and the porosity measurements in the dataset of Test#5. All the training and adaptive refinement parameters of GM-PNN were the same as those in Section 4.3.

The porosity status classification of the cascaded model of CNN and GM-PNN, denoted as CNN+PNN, is presented in Figure 5-14, with misclassified samples indicated by ‘x’ markers. It can be seen that the CNN+PNN model correctly classified most of the data samples to detect porosity occurrence. The performance of the CNN+PNN model is compared with the baseline CNN model in Table 5-14, Figure 5-15 and Figure 5-16. For porosity status classification, the fusion of CNN with GM-PNN remarkably improved the classification accuracy from 91.2% to 93.6%. The number of misclassified samples was reduced from 31 to 18. More importantly, as shown in Figure 5-16, the misclassified samples of CNN+PNN (all below 50 μm) were mainly those with very small pores (less than 10 μm). Compared with the baseline CNN, the CNN+PNN model had a much more reliable porosity detection for the pores above 10 μm, given that there

were 16 samples in the 10-50 µm range misclassified by the CNN model while only 4 were misclassified by the CNN+PNN model. Therefore, it can be said that by better considering the uncertainties in the melt-pool features, which originated from the noises in melt-pool images, the GM-PNN significantly improved the CNN's sensitivity to the pores with a sensible size (larger than 10 µm). For the pores below 10 µm, since they could not make sufficient impact on the melt pool and there are no sufficient detectable patterns in the melt-pool data, no matter what learning technique is used, the classification accuracy could not be further improved.

Table 5-14 Comparison of porosity monitoring model performance

	Porosity Classification Accuracy	*Local Volume Porosity Prediction*			
		R^2	*RMSE*	*NLL*	*Interval Width*
CNN	91.2 %	0.907	1.25%	N/A	N/A
CNN+PNN	93.6 %	0.950	0.92%	1.30	95%: 3.55% 99%: 4.71%

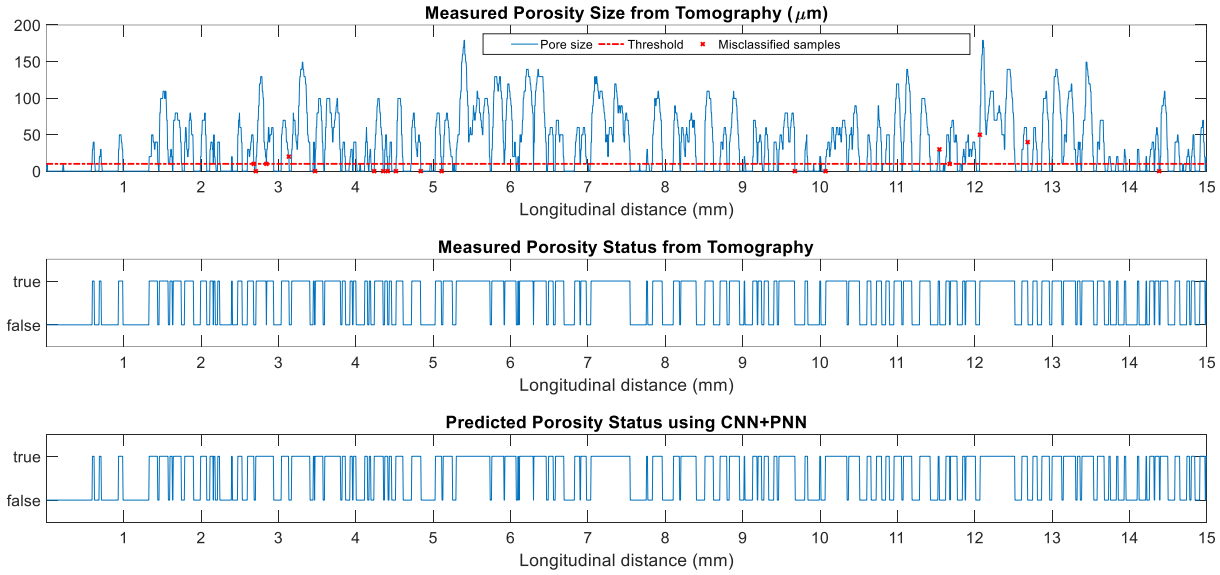
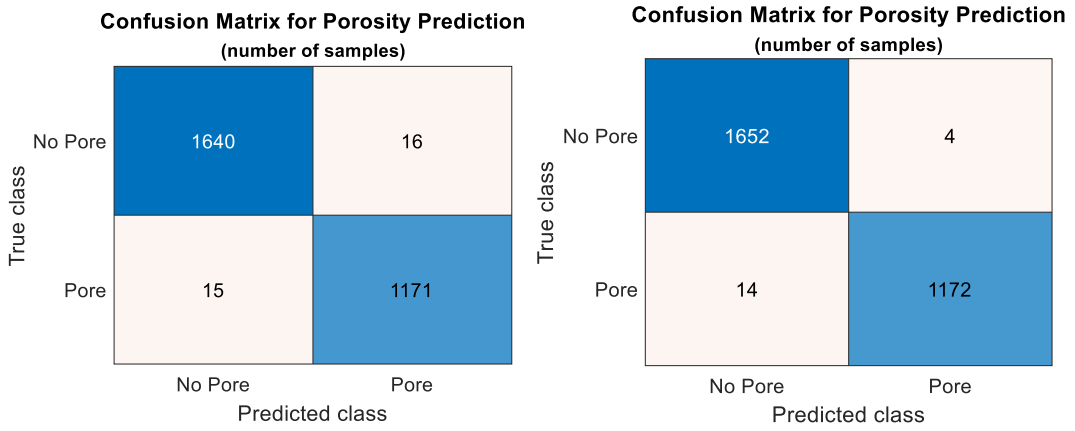


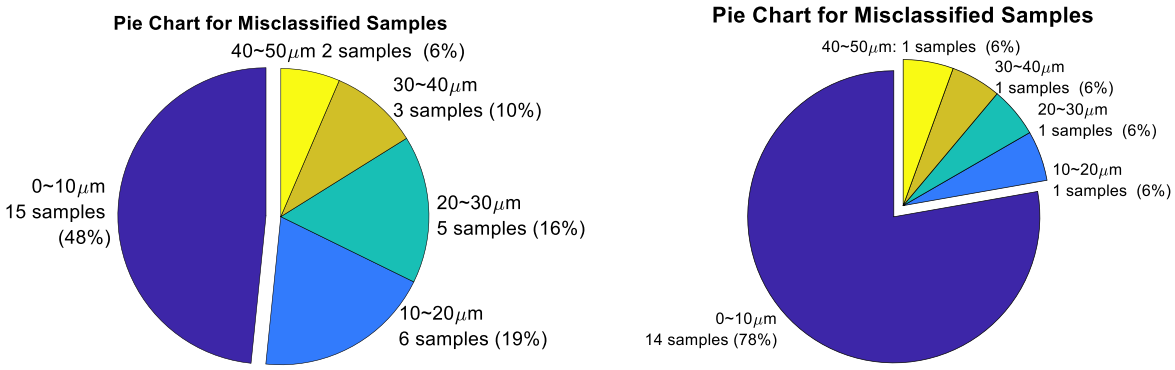
Figure 5-14 Classification of porosity status using CNN+PNN

Top: pore size extracted from the tomography measurement data; Middle: measured porosity status using the 10 µm threshold; Bottom: CNN+PNN classification results



(a) CNN (b) CNN+PNN

Figure 5-15 Confusion matrices of porosity status prediction



(a) CNN (b) CNN+PNN

Figure 5-16 Pie charts of misclassified samples for porosity status prediction

5.2.3.3 Results of Volume Porosity Prediction

Other than the detection of individual pores, for the AM deposition of materials, it is usually also of interest to know the volume porosity (ratio of porosity volume to the deposition volume) in the deposited parts, because the level of porosity is a key quality attribute that will influence the parts' performance such as mechanical strength. Test#1~4 in Table 5-10 inspected via cross-sectioning method were used to study the prediction of volume porosity, based on the assumption that though the pores are scattered inside a deposited track, the level of porosity may not be varying significantly across the track, so that the volume porosity evaluated from a set of cross-sections could represent the true porosity level in the track. Table 5-15 summarizes the volume porosity measured from multiple cross-sections for the four tests, in which the overall volume porosity is

calculated as the average ± standard deviation of the cross-section volume porosities. It can be seen that the variance of volume porosities among cross-sections is in an acceptable range compared with the mean magnitude of porosity, and thus the above assumption is reasonable, though not strict.

Table 5-15 The volume porosity measured from cross-sections
(Test#1 and #2 had only two cross-sections inspected)

Test#	*Volume porosity on cross-sections*			*Overall volume porosity*
	#1	#2	#3	
1	17.89%	11.99%	N/A	14.9±4.2%
2	12.08%	9.27%	N/A	10.7±2.0%
3	6.65%	9.86%	8.16%	8.2±1.6%
4	5.13%	4.51%	8.15%	5.9±1.9%

For the volume porosity model, instead of predicting the overall volume porosity in Table 5-15, which is averaged over the whole track, the melt-pool image at each longitudinal position is used to predict the local volume porosity in the neighborhood of that position. The evaluation of local volume porosity is illustrated in Figure 5-17, taking the 1st cross-section of Test#1 as an example. The pore size at each longitudinal position is added up and divided by the depth of track to obtain the volume porosity at that exact position, then a two-sided moving average filter sliding longitudinally with selected window size is applied to compute the local volume porosity. The volume porosities evaluated for different cross-sections of the same specimen are averaged and then used to label the melt-pool images from that experiment. Obviously, the window size of the moving average filter is an important hyperparameter in processing data for the monitoring model. Using a too-small window will make the local volume porosity too sensitive to the occurrence of individual pores while using a too-large window will fail to capture the local fluctuation of porosity and a local melt-pool image may not be suitable to predict the porosity over a too large span. In this study, by evaluating the performance of CNN models using another cross-validation analysis, it was found that a 2 mm moving average window achieved the best performance. Also, it was found that the number of convolutional filters could be reduced without degrading the performance, probably because predicting the volume porosity was less dependent on the detailed local behavior of the melt pool than detecting the occurrence of individual pores. Hence, the number of filters in Table 5-11 was reduced to 12, 16, 24, 24, and 24 for the five layers respectively, which made the

CNN model much more compact.

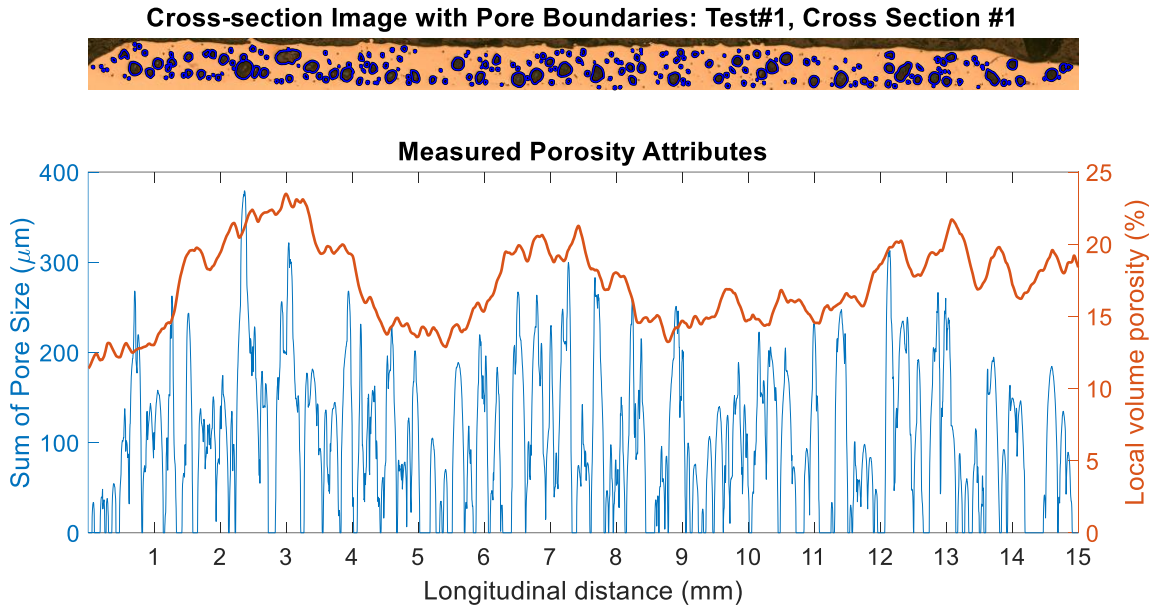


Figure 5-17 The evaluation of local volume porosity from a longitudinal cross-section

Top: the cross-section image with extracted pore boundaries highlighted; Bottom: pore size measured from the cross-section and local volume porosity evaluated using a 2 mm moving average window

With the calculated volume porosity and updated CNN structure, the CNN and CNN+PNN models were trained for the local volume porosity prediction task respectively. As shown in Table 5-14, the CNN+PNN model considerably outperformed the CNN model in this regression task with a higher R^2 and lower RMSE. Figure 5-18 shows the prediction for the experimental sample of Test#2. As can be seen, the prediction of local volume porosity by CNN was quite noisy due to the fluctuations in input images. In contrast, the prediction of CNN+PNN was much smoother because the GM-PNN module could quantify noisy fluctuations as uncertainty and generate confidence intervals (CI) from its predicted distributions to enclose them.

Finally, to validate the GM-PNN's capacity of modeling arbitrary types of uncertainties, the CNN+PNN model was used to predict the overall distribution of volume porosity for the same experimental sample in Figure 5-18, and the result is shown in Figure 5-19. This PDF prediction was generated by extracting the feature vectors belonging to this sample from the last convolutional layer, fitting the distribution of features as a 6-component Gaussian mixture, and

then propagating the fitted input uncertainty through the GM-PNN module. It can be seen that the predictive distribution of GM-PNN, as a Gaussian mixture, matched the true measurement histogram, even though the true distribution didn't belong to any specific distribution form. The ability to use Gaussian mixture to construct generic non-Gaussian predictive distributions is one of the advantages of the proposed GM-PNN over other models that could only provide Gaussian predictive distributions, such as the deep Gaussian processes and BNNs under Gaussian posterior assumptions.

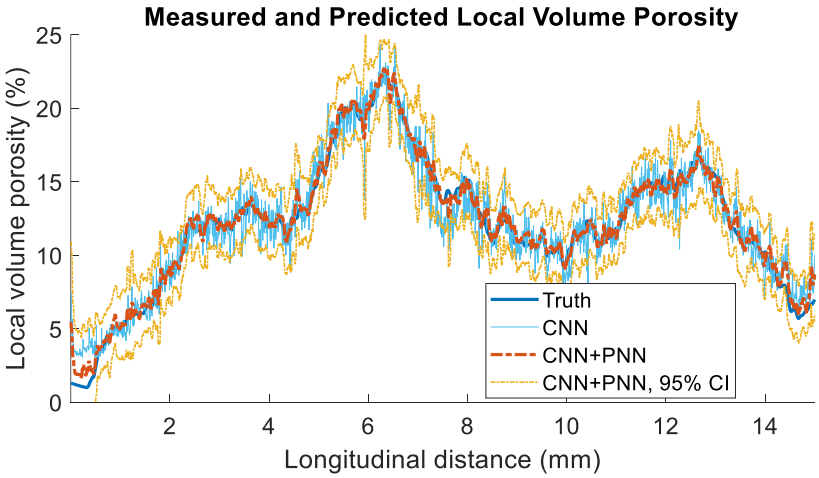


Figure 5-18 Predictions for local volume porosity

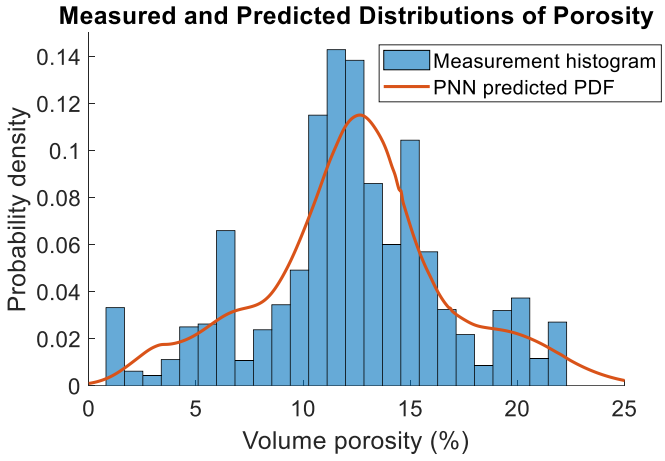


Figure 5-19 Prediction of the overall distribution of local volume porosity by CNN+PNN

5.3 Summary

This chapter describes the applications of the probabilistic neural network in Chapter 4 on two manufacturing process monitoring schemes. First, a robust tool wear monitoring scheme for turning processes was developed using a minimally-intrusive and low-cost instrumentation setup. A systematic feature normalization and selection procedure was proposed to eliminate the signal features' dependence on cutting conditions, cutting tools and workpiece materials and select the features that have the best performance in predicting tool wear. In particular, the feature fusion in the feature selection procedure was shown to be a methodical approach that was capable of finding combinations of multiple frequency bands and multiple frequency-domain features with maximum correlation to tool wear. The tool wear models were trained with two methods: one is the type-2 fuzzy network for interval uncertainty quantification and the other is the GM-PNN proposed in Chapter 4 for probabilistic uncertainty quantification. Experimental results show that by modeling the uncertainties probabilistically, the GM-PNN not only achieved higher accuracy of tool wear prediction, but also provided less conservative but still reliable tool change requests to prevent premature tool failure.

Second, a porosity monitoring scheme for laser AM processes was developed. A high-speed digital camera was used for in-process melt-pool sensing and a series of image processing tools were developed to precisely label each collected melt-pool image with measured porosity attributes. A compact CNN structure was designed to directly learn melt-pool features to predict porosity. For both the classification task of porosity detection and the regression task of local volume porosity prediction, the classical CNN models were compared with the cascade models of CNN and GM-PNN. Experimental results reveal that the CNN models alone had achieved exemplary capacity in detecting micro-porosity below 50 μm while the monitoring works in the literature are mostly addressing the pores above 100 μm. The fusion of CNN with GM-PNN made the performance even better by not only improving the porosity prediction accuracy but also offering essential uncertainty information. Especially, the non-Gaussian distribution of local volume porosity could be accurately predicted with the GM-PNN module. Based on the results of the two monitoring schemes, the GM-PNN proves to be effective in quantifying uncertainty and facilitating reliable decision-making in the manufacturing environment. Therefore, it will be a powerful solution to address practical problems subject to significant uncertainties.

6. CONCLUSIONS AND FUTURE WORK

In this study, a series of high-fidelity and computationally efficient uncertainty analysis approaches were developed for data-driven neural network models and their applications on two manufacturing process monitoring problems were presented:

1. A nonlinear uncertainty propagation scheme based on adaptive refinement of Gaussian mixtures was proposed. The Gaussian mixture was used to characterize the general-form probability distributions of uncertainties, which yielded higher fidelity than presumed distribution forms, and was more compact than Monte Carlo sampling. Through active assessment of nonlinearity and adaptive splitting of distorted Gaussian components, this scheme could effectively propagate uncertainties in arbitrary forms through a neural network. Its accuracy was comparable to the large-scale Monte Carlo sampling while its computation time was much shorter. Therefore, it could be used as a versatile scheme to address the uncertainty propagation in recurrent and deep neural networks.
2. An adaptive Gaussian mixture filter (AGMF) for Bayesian state estimation was derived. By approximating the dynamics of a nonlinear system with a feedforward neural network and applying the adaptive Gaussian mixture refinement, high-fidelity state prediction was achieved. Subsequently, the refinement scheme was extended with a likelihood divergence criterion to assess the nonlinearity in Bayesian measurement update so that the PDF of unmeasurable states could be closely tracked over time. It was proved that under mild conditions, the L^1 norm error of state PDF estimation can be bounded to a minimal level. Based on the testing on a series of challenging nonlinear filtering problems, the AGMF proved to be an efficient solution for highly nonlinear state estimation applications.
3. A probabilistic neural network with Gaussian-mixture-distributed parameters (GM-PNN) was developed. In addition to the adaptive refinement by assessing nonlinearity, a new criterion was introduced to ensure the fidelity of probabilistic linear transformation, so that the predictive output distribution could be inferred accurately without sampling or integration. The gradients of loss function with respect to the probabilistic parameters were all derived explicitly, and thus the GM-PNN can be trained with any sampling-free backpropagation method. The GM-PNN proved its robust learning capacity from noisy

data by achieving a state-of-the-art accuracy on a series of benchmark datasets.

4. Two comparative studies of manufacturing process monitoring were conducted. A tool wear monitoring scheme for turning processes was built based on engineered signal features, in which the GM-PNN was compared with a type-2 fuzzy network. Likewise, a porosity monitoring system for laser AM processes was built by direct feature learning from melt-pool data, in which the CNN models with and without embedded GM-PNN module were compared. In both schemes, the GM-PNN not only remarkably improved the prediction accuracy of process conditions but also offered more effective uncertainty quantification needed by reliable process-level decision-making and intervention.

Based on the developed uncertainty analysis methods and their proven successes in practical applications, some directions for future studies are suggested as below:

1. Closed-loop control system synthesis based on AGMF. The AGMF models the system dynamics as a neural network and thus a data-model-based controller design method can be applied without extra modeling effort. The AGMF also offers an accurate estimation of unmeasurable states with quantified error bounds, which makes it possible to consider the high-fidelity feedback state uncertainty in the controller design step, though how to ensure the stability and robustness of such a closed-loop system needs further studies.
2. Parameter estimation using AGMF. Using Kalman filters, or its nonlinear variants, to train data-driven models, like recurrent networks, has been widely studied. Though in this study the AGMF is proposed as a nonlinear state estimator, it can be extended to the parameter estimation problems in training neural networks. Its capacity of methodically addressing the nonlinearity and uncertainty in the model will be invaluable in advancing the performance of existing filter-based training methods.
3. Probabilistic deep and recurrent neural networks. Though the GM-PNN in this study is mainly formulated as a multi-layer feedforward network, due to its good performance, it is of interest to expand its inference and training method to more complicated models. However, this will introduce computational bottlenecks to be addressed. For example, though the convolutional layer in CNN is theoretically a linear operation, the uncertainty propagation through it will involve SVD and matrix multiplications with up to several thousand or million dimensions. Therefore, the appropriate simplification for uncertainty-related computations in high-dimensional layers needs to be studied.

APPENDIX A. DERIVATION OF THE KL DIVERGENCE FOR NONLINEARITY DETECTION

To derive the KL divergence evaluation in Eq. (2.9), substitute Eq. (2.8) into Eq. (2.6) and then by using the rule change of variables, the integral variables in Eq. (2.6) can be changed from the post-activation states $\mathbf{s}=\left[s_1,s_2,\ldots,s_k\right]^T$ to pre-activation states $\mathbf{z}=\left[z_1,z_2,\ldots,z_k\right]^T$:

$$
\begin{aligned}
D_{KL}\left(p\,\|\,\hat{p}\right) &= \int_{-\infty}^{\infty}\left|\frac{d\mathbf{z}}{d\mathbf{s}}\right|\mathcal{N}\left(\mathbf{f}^{-1}\left(\mathbf{s}\right)|\,\boldsymbol{\mu},\boldsymbol{\Sigma}\right)\log\left(\left|\frac{d\mathbf{z}}{d\mathbf{s}}\right|\frac{\mathcal{N}\left(\mathbf{f}^{-1}\left(\mathbf{s}\right)|\,\boldsymbol{\mu},\boldsymbol{\Sigma}\right)}{\mathcal{N}\left(\mathbf{s}\,|\,\mathbf{f}\left(\boldsymbol{\mu}\right),\mathbf{A}\boldsymbol{\Sigma}\mathbf{A}^T\right)}\right)d\mathbf{s}\\
&= \int_{-\infty}^{\infty}\mathcal{N}\left(\mathbf{z}\,|\,\boldsymbol{\mu},\boldsymbol{\Sigma}\right)\log\left(\left|\frac{d\mathbf{f}\left(\mathbf{z}\right)}{d\mathbf{z}}\right|\frac{\mathcal{N}\left(\mathbf{z}\,|\,\boldsymbol{\mu},\boldsymbol{\Sigma}\right)}{\mathcal{N}\left(\mathbf{f}\left(\mathbf{z}\right)|\,\mathbf{f}\left(\boldsymbol{\mu}\right),\mathbf{A}\boldsymbol{\Sigma}\mathbf{A}^T\right)}\right)d\mathbf{z}\\
&= \int_{-\infty}^{\infty}\mathcal{N}\left(\mathbf{z}\,|\,\boldsymbol{\mu},\boldsymbol{\Sigma}\right)\log\mathcal{N}\left(\mathbf{z}\,|\,\boldsymbol{\mu},\boldsymbol{\Sigma}\right)d\mathbf{z}+\int_{-\infty}^{\infty}\mathcal{N}\left(\mathbf{z}\,|\,\boldsymbol{\mu},\boldsymbol{\Sigma}\right)\log\left|\frac{d\mathbf{f}\left(\mathbf{z}\right)}{d\mathbf{z}}\right|d\mathbf{z}\\
&\quad-\int_{-\infty}^{\infty}\mathcal{N}\left(\mathbf{z}\,|\,\boldsymbol{\mu},\boldsymbol{\Sigma}\right)\log\mathcal{N}\left(\mathbf{f}\left(\mathbf{z}\right)|\,\mathbf{f}\left(\boldsymbol{\mu}\right),\mathbf{A}\boldsymbol{\Sigma}\mathbf{A}^T\right)d\mathbf{z}
\end{aligned}
\tag{A.1}
$$

The first term in Eq. (A.1) is the differential entropy of a Gaussian distribution, which has a closed-form solution [216]:

$$
\int_{-\infty}^{\infty}\mathcal{N}\left(\mathbf{z}\,|\,\boldsymbol{\mu},\boldsymbol{\Sigma}\right)\log\mathcal{N}\left(\mathbf{z}\,|\,\boldsymbol{\mu},\boldsymbol{\Sigma}\right)d\mathbf{z}=-\frac{k}{2}-\frac{1}{2}\log\left(\left(2\pi\right)^k\left|\boldsymbol{\Sigma}\right|\right)
\tag{A.2}
$$

Since the activation functions considered in this work are scaler mappings, the Jacobian matrix is always diagonal:

$$
\begin{aligned}
&s_i=f\left(z_i\right)\quad i=1,2,\ldots,k\\
&\frac{ds_i}{z_j}=\begin{cases}f'\left(z_j\right) & \text{if } i=j\\ 0 & \text{if } i\neq j\end{cases}\\
&\left|\frac{d\mathbf{f}\left(\mathbf{z}\right)}{d\mathbf{z}}\right|=\prod_{i=1}^{k}f'\left(z_i\right)\\
&\mathbf{A}=\begin{bmatrix}f'\left(\mu_1\right) & & \\ & \ddots & \\ & & f'\left(\mu_k\right)\end{bmatrix}\quad\left|\mathbf{A}\right|=\prod_{i=1}^{k}f'\left(\mu_i\right)
\end{aligned}
\tag{A.3}
$$

Therefore the second term in Eq. (A.1) can be expressed using Eq. (A.3) as:

$$
\begin{aligned}
&\int_{-\infty}^{\infty} \mathcal{N}\left(\mathbf{z} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}\right) \log \left| \frac{d\mathbf{f}\left(\mathbf{z}\right)}{d\mathbf{z}} \right| d\mathbf{z} \\
&= \int_{-\infty}^{\infty} \mathcal{N}\left(\mathbf{z} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}\right) \log \prod_{i=1}^{k} f'\left(z_i\right) d\mathbf{z} \\
&= \sum_{i=1}^{k} \int_{-\infty}^{\infty} \mathcal{N}\left(z_i \mid \mu_i, \sigma_i^{\,2}\right) \log\left(f'\left(z_i\right)\right) dz_i \\
&= \sum_{i=1}^{k} E_{p(z_i)}\left[\log\left(f'\left(z_i\right)\right)\right]
\end{aligned}
\tag{A.4}
$$

which is the sum of expectations of the logarithm of activation function derivatives. The third term in Eq. (A.1) can be expanded as:

$$
\begin{aligned}
&\int_{-\infty}^{\infty} \mathcal{N}\left(\mathbf{z} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}\right) \log \mathcal{N}\left(\mathbf{f}\left(\mathbf{z}\right) \mid \mathbf{f}\left(\boldsymbol{\mu}\right), \mathbf{A}\boldsymbol{\Sigma}\mathbf{A}^{T}\right) d\mathbf{z} \\
&= \int_{-\infty}^{\infty} \mathcal{N}\left(\mathbf{z} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}\right) \log \left(\frac{1}{\sqrt{\left(2\pi\right)^{k} \left|\boldsymbol{\Sigma}\right| \left|\mathbf{A}\right|^{2}}} e^{-\frac{1}{2}\left(\mathbf{f}\left(\mathbf{z}\right)-\mathbf{f}\left(\boldsymbol{\mu}\right)\right)^{T}\left(\mathbf{A}\boldsymbol{\Sigma}\mathbf{A}^{T}\right)^{-1}\left(\mathbf{f}\left(\mathbf{z}\right)-\mathbf{f}\left(\boldsymbol{\mu}\right)\right)} \right) d\mathbf{z} \\
&= -\frac{1}{2} \log\left(\left(2\pi\right)^{k} \left|\boldsymbol{\Sigma}\right|\right) - \log\left|\mathbf{A}\right| \\
&-\frac{1}{2} \int_{-\infty}^{\infty} \mathcal{N}\left(\mathbf{z} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}\right) \left(\mathbf{f}\left(\mathbf{z}\right)-\mathbf{f}\left(\boldsymbol{\mu}\right)\right)^{T} \left(\mathbf{A}\boldsymbol{\Sigma}\mathbf{A}^{T}\right)^{-1} \left(\mathbf{f}\left(\mathbf{z}\right)-\mathbf{f}\left(\boldsymbol{\mu}\right)\right) d\mathbf{z}
\end{aligned}
\tag{A.5}
$$

In Eq.(A.5), the first term is a constant that cancels the second term in the differential entropy in Eq. (A.2). Using Eq. (A.3), the second term can be computed as $\log\left|\mathbf{A}\right| = \sum_{i=1}^{k} \log\left(f'\left(\mu_i\right)\right)$. Then the integrand in the third item can be rewritten as:

$$
\begin{aligned}
&\left(\mathbf{f}\left(\mathbf{z}\right)-\mathbf{f}\left(\boldsymbol{\mu}\right)\right)^{T} \left(\mathbf{A}\boldsymbol{\Sigma}\mathbf{A}^{T}\right)^{-1} \left(\mathbf{f}\left(\mathbf{z}\right)-\mathbf{f}\left(\boldsymbol{\mu}\right)\right) \\
&= \sum_{i=1}^{k} \sum_{j=1}^{k} \frac{c_{ij}\left(f\left(x_i\right)-f\left(\mu_i\right)\right)\left(f\left(x_j\right)-f\left(\mu_j\right)\right)}{f'\left(\mu_i\right) f'\left(\mu_j\right)}
\end{aligned}
\tag{A.6}
$$

Given that

$$
\begin{aligned}
&\mathbf{f}\left(\mathbf{z}\right)-\mathbf{f}\left(\boldsymbol{\mu}\right) = \begin{bmatrix} f\left(z_1\right)-f\left(\mu_1\right) & \cdots & f\left(z_k\right)-f\left(\mu_k\right) \end{bmatrix}^{T} \\
&\left(\mathbf{A}\boldsymbol{\Sigma}\mathbf{A}^{T}\right)^{-1} = \mathbf{A}^{-1}\boldsymbol{\Sigma}^{-1}\mathbf{A}^{-1} = \left[c_{ij} \Big/ f'\left(\mu_i\right) f'\left(\mu_j\right) \right]_{i,j=1,\ldots,k}
\end{aligned}
$$

where c_{ij} is the element in the i-th row and j-th column in the inverse of covariance matrix $\boldsymbol{\Sigma}^{-1}$. The third term in Eq.(A.5) can be evaluated as:

$$\int_{-\infty}^{\infty} p(\mathbf{z})\left(\mathbf{f}(\mathbf{z})-\mathbf{f}(\boldsymbol{\mu})\right)^{T}\left(\mathbf{A}\boldsymbol{\Sigma}\mathbf{A}^{T}\right)^{-1}\left(\mathbf{f}(\mathbf{z})-\mathbf{f}(\boldsymbol{\mu})\right)d\mathbf{z}$$

$$=\sum_{i=1}^{k}\sum_{j=1}^{k}\left(\frac{c_{ij}}{f'(\mu_i)f'(\mu_j)}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}p(z_i,z_j)\left(f(z_i)-f(\mu_i)\right)\left(f(z_j)-f(\mu_j)\right)dz_i dz_j\right) \quad \text{(A.7)}$$

$$=\sum_{i=1}^{k}\sum_{j=1}^{k}\left(\frac{c_{ij}}{f'(\mu_i)f'(\mu_j)}E_{p(z_i,z_j)}\left[\left(f(z_i)-f(\mu_i)\right)\left(f(z_j)-f(\mu_j)\right)\right]\right)$$

Substitute Eq. (A.7) into Eq. (A.5) and then substitute Eq. (A.2), (A.4) and (A.5) into Eq. (A.1), the Eq. (2.9) can be obtained.

APPENDIX B. THE QUADROTOR DRONE DYNAMICS

A description of the dynamics of the quadrotor drone used in Section 2.3.3 is provided below, which is adopted from [210]. As mentioned in Eq. (2.27), the drone system considered in this work has 12 state variables and 4 inputs:

$$\mathbf{x} = \begin{bmatrix} x & y & z & \dot{x} & \dot{y} & \dot{z} & \phi & \theta & \psi & p & q & r \end{bmatrix}^T$$
$$\mathbf{u} = \begin{bmatrix} \omega_1 & \omega_2 & \omega_3 & \omega_4 \end{bmatrix}^T$$

where the first 6 states are *x*, *y*, *z* locations of the drone in the inertial frame and their velocities, φ, θ, ψ are the pitch, roll and yaw angles of the drone in inertial frame and *p*, *q*, *r* are the angular velocities in the body frame. The four inputs in **u** are angular velocities of the four propellers. The coordinate frames of the drone are illustrated in Figure B-1.

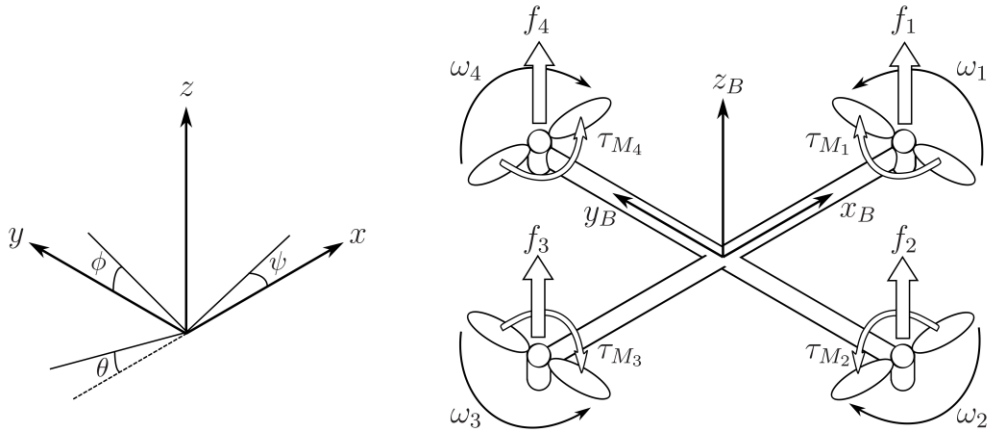


Figure B-1 The inertial and body frames of a quadrotor drone

With the coordinate transformation from the body frame to the inertial frame, the acceleration of the drome, as derived in [210], can be calculated as:

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = -g \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} + \frac{T}{m} \begin{bmatrix} \cos\psi \sin\theta \cos\phi + \sin\psi \sin\phi \\ \sin\psi \sin\theta \cos\phi - \cos\psi \sin\phi \\ \cos\theta \cos\phi \end{bmatrix} - \begin{bmatrix} A_x & 0 & 0 \\ 0 & A_y & 0 \\ 0 & 0 & A_z \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} \tag{B.1}$$

where the *g* is the gravitational acceleration, *m* is the mass of the drone, *T* is the total thrust force, and A_x, A_y, A_z are the drag force coefficients for velocities in the corresponding directions of the

inertial frame to consider the aerodynamical effect.

The angular acceleration of the drone in body frame can be written as:

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} \left(I_{yy} - I_{zz}\right) qr / I_{xx} \\ \left(I_{zz} - I_{xx}\right) pr / I_{yy} \\ \left(I_{xx} - I_{yy}\right) pq / I_{zz} \end{bmatrix} + \begin{bmatrix} \tau_{\phi} / I_{xx} \\ \tau_{\theta} / I_{yy} \\ \tau_{\psi} / I_{zz} \end{bmatrix} \tag{B.2}$$

where I_{xx}, I_{yy}, I_{zz} are the moments of inertia of the drone in the corresponding directions, and τ_{φ}, τ_{θ}, τ_{ψ} are the torques around pitch, roll and yaw angles.

The transformation of angular velocities from the body frame to the inertial frame is:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi/\cos\theta & \cos\phi/\cos\theta \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{B.3}$$

The thrust forces f of the four propellers and torques τ around propeller axes are determined by their angular velocities ω:

$$f_i = k\omega_i^2 \quad \tau_i = b\omega_i^2 \quad i = 1, \ldots, 4 \tag{B.4}$$

where k and b are the lift constant and drag constants, respectively.

The total thrust force T and the torques around pitch, roll and yaw angles (τ_{φ}, τ_{θ}, τ_{ψ}) are:

$$\begin{aligned} T &= \sum_{i=1}^{4} f_i = k \sum_{i=1}^{4} \omega_i^2 \\ \tau_{\phi} &= lk\left(-\omega_2^2 + \omega_4^2\right) \\ \tau_{\theta} &= lk\left(-\omega_1^2 + \omega_3^2\right) \\ \tau_{\psi} &= \sum_{i=1}^{4} \tau_i = b \sum_{i=1}^{4} \omega_i^2 \end{aligned} \tag{B.5}$$

where l is the distance between the rotor and the center of mass of the drone.

Combining Eq. (B.1)-(B.5), the equation of motion of the drone can be written as:

$$\dot{\mathbf{x}} = \mathbf{f}\left(\mathbf{x}, \mathbf{u}\right) = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \frac{k\sum_{i=1}^{4}\omega_i^2}{m}\left(\sin\phi\sin\psi + \cos\phi\cos\psi\sin\theta\right) - \frac{A_x}{m}\dot{x} \\ \frac{k\sum_{i=1}^{4}\omega_i^2}{m}\left(\sin\phi\cos\psi - \cos\phi\sin\psi\sin\theta\right) - \frac{A_y}{m}\dot{y} \\ \frac{k\sum_{i=1}^{4}\omega_i^2}{m}\cos\phi\cos\theta - g - \frac{A_z}{m}\dot{z} \\ q\frac{\sin\phi}{\cos\theta} + r\frac{\cos\phi}{\cos\theta} \\ q\cos\phi - r\sin\phi \\ p + q\sin\phi\tan\theta + r\cos\phi\tan\theta \\ \frac{I_y - I_z}{I_x}qr + \frac{1}{I_x}lk\left(-\omega_2^2 + \omega_4^2\right) \\ \frac{I_z - I_x}{I_y}pr + \frac{1}{I_y}lk\left(-\omega_1^2 + \omega_3^2\right) \\ \frac{I_x - I_y}{I_z}pq + \frac{1}{I_z}b\sum_{i=1}^{4}\omega_i^2 \end{bmatrix} \tag{B.6}$$

The above equation of motion is used for the simulations in Section 2.3.3 to generate Monte Carlo samples and training samples for the recurrent neural network. The drone parameters used for simulation are the same as those in [210]:

Table B-1 Drone parameter values for simulation

Parameter	*Value*	*Unit*	*Parameter*	*Value*	*Unit*
g	9.81	m/s^2	I_{xx}	4.856×10^3	$kg\times m^2$
m	0.468	kg	I_{yy}	4.856×10^3	$kg\times m^2$
l	0.225	m	I_{zz}	8.801×10^3	$kg\times m^2$
k	2.980×10^6	$kg\times m$	A_x	0.25	kg/s
b	1.140×10^7	$kg\times m^2$	A_y	0.25	kg/s
			A_z	0.25	kg/s

APPENDIX C. PROOF OF LEMMA 1

This proof considers a pre-activation PDF with a single Gaussian density $p(\mathbf{z})=\mathcal{N}(\mathbf{z}|\boldsymbol{\mu}_z,\boldsymbol{\Sigma}_z)$. The terms in Eq. (3.17) can be clustered into two groups that converge respectively:

$$\begin{aligned} &D_{KL}\left(p(\mathbf{s})\,\|\,\hat{p}(\mathbf{s})\right)=D_1+D_2\le|D_1|+|D_2| \quad \text{with} \\ &D_1=\log|\mathbf{A}|-E_{p(\mathbf{z})}\left[\log\left|\phi'(\mathbf{z})\right|\right] \;\text{ and }\; D_2=\frac{1}{2}E_{p(\mathbf{z})}\left[\mathcal{F}(\mathbf{z})\right]-\frac{m}{2} \end{aligned} \tag{C.1}$$

In D_1, given that the Jacobian matrix of sigmoid function is diagonal, the expectation can be written as $E_{p(\mathbf{z})}\left[\log\left|\phi'(\mathbf{z})\right|\right]=E_{p(\mathbf{z})}\left[\prod_{i=1}^{m}\log\left(\phi'(z_i)\right)\right]=\sum_{i=1}^{m}E_{p(z_i)}\left[\log\left(\phi'(z_i)\right)\right]$, and thus:

$$\begin{aligned} |D_1|&=\left|\sum_{i=1}^{m}\left\{\log\left(\phi'(\mu_i)\right)-E_{p(z_i)}\left[\log\left(\phi'(z_i)\right)\right]\right\}\right| \\ &\le\sum_{i=1}^{m}E_{p(z_i)}\left|\log\left(\phi'(x_i)\right)-\log\left(\phi'(\mu_i)\right)\right| \end{aligned} \tag{C.2}$$

For logistic sigmoid function $\phi(z_i)=1/\left(1+e^{-z_i}\right)$, it can be shown that $\left|d\log\left(\phi'(z_i)\right)/dz_i\right|=\left|\left(1-e^{-z_i}\right)/\left(1+e^{-z_i}\right)\right|\le 1$ and $\left|\log\left(\phi'(z_i)\right)-\log\left(\phi'(\mu_i)\right)\right|\le L_a|z_i-\mu_i|$ with L_a=1. Using the fact that for $z_i\sim\mathcal{N}(\mu_i,\,\sigma_i^2)$, $E_{p(z_i)}|z_i-\mu_i|=\sqrt{2\sigma_i^2/\pi}$ and $\sum\sqrt{\sigma_i^2}\le 2\sqrt{\sum\sigma_i^2}$, it can be obtained that:

$$\begin{aligned} |D_1|&\le L_\sigma\sum_{i=1}^{m}\left\{E_{p(z_i)}|z_i-\mu_i|\right\}=L_a\sum_{i=1}^{m}\sqrt{2\sigma_i^2/\pi} \\ &\le 2L_a\sqrt{2\sum_{i=1}^{m}\sigma_i^2/\pi}=2L_a\sqrt{2\,\mathrm{tr}(\boldsymbol{\Sigma}_z)/\pi} \end{aligned} \tag{C.3}$$

For D_2, using the Taylar's expansion $\phi(\mathbf{z})=\phi(\boldsymbol{\mu}_z)+\mathbf{A}(\mathbf{z}-\boldsymbol{\mu}_z)+\mathbf{r}_2(\mathbf{z})$ with a second-order remainder $\mathbf{r}_2(\mathbf{z})=0.5\left[\phi''(\xi_1)(z_1-\mu_1)^2\quad\ldots\quad\phi''(\xi_m)(z_m-\mu_m)^2\right]^T$, the $\mathcal{F}(\mathbf{z})$ can be rewritten as $\mathcal{F}(\mathbf{z})=(\mathbf{z}-\boldsymbol{\mu}_z)^T\,\boldsymbol{\Sigma}_z^{-1}(\mathbf{z}-\boldsymbol{\mu}_z)+2(\mathbf{z}-\boldsymbol{\mu}_z)^T\,\boldsymbol{\Sigma}_z^{-1}\mathbf{A}^{-1}\mathbf{r}_2+\mathbf{r}_2^T\mathbf{A}^{-T}\boldsymbol{\Sigma}_z^{-1}\mathbf{A}^{-1}\mathbf{r}_2$. Based on the expectation of quadratic function (Results 8.5, page. 170 in [220]), the first term can be evaluated as:

$$E_{p(\mathbf{z})=\mathcal{N}(\boldsymbol{\mu}_z,\boldsymbol{\Sigma}_z)}\left[(\mathbf{z}-\boldsymbol{\mu}_z)^T\,\boldsymbol{\Sigma}_z^{-1}(\mathbf{z}-\boldsymbol{\mu}_z)\right]=\mathrm{tr}\left(\boldsymbol{\Sigma}_z\boldsymbol{\Sigma}_z^{-1}\right)=m \tag{C.4}$$

which cancels the $m/2$ term in D_2. Using the SVD in Eq. (2.11) and $\boldsymbol{\Sigma}_z^{-1}=\mathbf{V}\boldsymbol{\Lambda}^{-1}\mathbf{V}^T$, the second term in $\mathcal{F}(\mathbf{z})$ is upper bounded by:

$$\begin{aligned}\left(\mathbf{z}-\boldsymbol{\mu}_z\right)^T \boldsymbol{\Sigma}_z{}^{-1}\mathbf{A}^{-1}\mathbf{r}_2\left(\mathbf{z}\right) &= \sum_{i=1}^{m}\frac{1}{\lambda_i}\left\langle \mathbf{v}_i, \mathbf{z}-\boldsymbol{\mu}_z\right\rangle\left\langle \mathbf{v}_i, \mathbf{A}^{-1}\mathbf{r}_2\right\rangle \\ &\le \sum_{i=1}^{m}\frac{1}{\lambda_i}\left\|\mathbf{v}_i\right\|_2{}^2\left\|\mathbf{z}-\boldsymbol{\mu}_z\right\|_2\left\|\mathbf{A}^{-1}\mathbf{r}_2\right\|_2 \le \frac{1}{\lambda_{\min}}\left\|\mathbf{z}-\boldsymbol{\mu}_z\right\|_2\left\|\mathbf{A}^{-1}\mathbf{r}_2\right\|_2\end{aligned} \tag{C.5}$$

where $\|\ \|_2$ denotes the 2-norm of a vector and for all the singular vectors $\|\mathbf{v}_i\|_2=1$. The expectation of the product of two norm terms is an inner product that satisfies the Cauchy–Schwarz inequality $E\left[XY\right]^2 \le E\left[X^2\right]E\left[Y^2\right]$, and the expectation of each norm can be evaluated as:

$$E_{p(\mathbf{z})}\left[\left\|\mathbf{z}-\boldsymbol{\mu}_z\right\|_2{}^2\right] = \sum\nolimits_{i=1}^{m} E_{p(z_i)}\left[\left(z_i-\mu_i\right)^2\right] = \mathrm{tr}\left(\boldsymbol{\Sigma}_z\right) \tag{C.6}$$

$$\begin{aligned}E_{p(\mathbf{z})}\left[\left\|\mathbf{A}^{-1}\mathbf{r}_2\left(\mathbf{z}\right)\right\|_2{}^2\right] &= \sum_{i=1}^{m} E_{p(z_i)}\left[\left(\frac{1}{2}\frac{\phi"\left(\xi_i\right)}{\phi'\left(\mu_i\right)}\right)^2\left(z_i-\mu_i\right)^4\right] \\ &\le L_b\sum_{i=1}^{m}3\sigma_i{}^4 \le 3L_b\left(\sum\nolimits_{i=1}^{m}\sigma_i{}^2\right)^2 = 3L_b\,\mathrm{tr}\left(\boldsymbol{\Sigma}_z\right)^2\end{aligned} \tag{C.7}$$

where $L_b = \left(\max\left|\phi"\right| \Big/ 2\min\left|\phi'\right|\right)^2$ and factor 3 is the kurtosis of the Gaussian variable z_i. For the logistic function, $\max\left|\phi"\left(\xi_i\right)\right| < 0.1$ and $\min\left|\phi'\left(\mu_i\right)\right| > \phi\left(M_\mu\right)\left(1-\phi\left(M_\mu\right)\right)$ given that $\max(\mu_i) < M_\mu$. Substituting Eq. (C.6) and (C.7) into the Cauchy–Schwarz inequality of Eq. (C.5) gives:

$$E_{p(\mathbf{z})}\left[\left(\mathbf{z}-\boldsymbol{\mu}_z\right)^T \boldsymbol{\Sigma}_z{}^{-1}\mathbf{A}^{-1}\mathbf{r}_2\right] \le \frac{\sqrt{3L_b\,\mathrm{tr}\left(\boldsymbol{\Sigma}_z\right)^3}}{\lambda_{\min}} \tag{C.8}$$

For the last term, using the same method as in Eq. (C.5) and (C.7), it can be obtained that:

$$E_{p(\mathbf{z})}\left[\mathbf{r}_2{}^T\mathbf{A}^{-T}\boldsymbol{\Sigma}_z{}^{-1}\mathbf{A}^{-1}\mathbf{r}_2\right] \le E_{p(\mathbf{z})}\left[\frac{\left\|\mathbf{A}^{-1}\mathbf{r}_2\right\|_2{}^2}{\lambda_{\min}}\right] \le \frac{3L_b\,\mathrm{tr}\left(\boldsymbol{\Sigma}_z\right)^2}{\lambda_{\min}} \tag{C.9}$$

Since the singular values satisfy $\lambda_{\max}/\lambda_{\min} < M_\lambda$, the trace has $\mathrm{tr}\left(\boldsymbol{\Sigma}_z\right)/\lambda_{\min} \le m\,\lambda_{\max}/\lambda_{\min} \le mM_\lambda$. Using Eq. (C.4), (C.8) and (C.9), an upper bound of D_2 is:

$$\left|D_2\right| \le mM_\lambda\sqrt{3L_b\,\mathrm{tr}\left(\boldsymbol{\Sigma}_z\right)} + 3mM_\lambda L_b\,\mathrm{tr}\left(\boldsymbol{\Sigma}_z\right) \tag{C.10}$$

with $L_b = 0.1\Big/\phi\left(M_\mu\right)\left(1-\phi\left(M_\mu\right)\right)$. Finally, substitute Eq. (C.10) and (C.3) into Eq. (C.1):

$$D_{KL}\left(p(\mathbf{s})\,\|\,\hat{p}(\mathbf{s})\right) \le b\sqrt{\mathrm{tr}\left(\boldsymbol{\Sigma}_z\right)} + a\,\mathrm{tr}\left(\boldsymbol{\Sigma}_z\right) \quad \text{with}$$
$$a = 3mM_{\lambda}L_b \quad \text{and} \quad b = 2L_a\sqrt{\frac{2}{\pi}} + mM_{\lambda}\sqrt{3L_b} \tag{C.11}$$

Therefore, for any positive th_D, there exists a positive root $\sigma_D = \sqrt{\frac{th_D}{a} + \left(\frac{b}{2a}\right)^2} - \frac{b}{2a} > 0$ such that if $\sqrt{\mathrm{tr}\left(\boldsymbol{\Sigma}_z\right)} \le \sigma_D$, the KL divergence is lower than th_D. For other sigmoid functions like the hyperbolic tangent, the constants L_a and L_b can also be derived and the Lemma 1 still holds.

APPENDIX D. PROOF OF LEMMA 3

This proof considers a prior state PDF with a single Gaussian density $p^{-}(\mathbf{x}_k)=\mathcal{N}(\mathbf{x}_k|\boldsymbol{\mu}_k^{-},\boldsymbol{\Sigma}_k^{-})$. Expanding the $\mathcal{G}(\mathbf{x}_k)$ and $\hat{\mathcal{G}}(\mathbf{x}_k)$ in Eq. (3.24), it can be obtained that:

$$\begin{aligned}
&\log\left(\frac{p(\mathbf{y}_k|\mathbf{x}_k)}{\hat{p}(\mathbf{y}_k|\mathbf{x}_k)}\right)=0.5G_1-0.5G_2+G_3\\
&G_1=\left(\mathbf{H}_k\left(\mathbf{x}_k-\boldsymbol{\mu}_k^{-}\right)\right)^T\mathbf{R}_k^{-1}\left(\mathbf{H}_k\left(\mathbf{x}_k-\boldsymbol{\mu}_k^{-}\right)\right)\\
&G_2=\left(\mathbf{g}(\mathbf{x}_k)-\mathbf{g}\left(\boldsymbol{\mu}_k^{-}\right)\right)^T\mathbf{R}_k^{-1}\left(\mathbf{g}(\mathbf{x}_k)-\mathbf{g}\left(\boldsymbol{\mu}_k^{-}\right)\right)\\
&G_3=\left(\mathbf{y}_k-\mathbf{g}\left(\boldsymbol{\mu}_k^{-}\right)\right)^T\mathbf{R}_k^{-1}\left(\mathbf{g}(\mathbf{x}_k)-\mathbf{g}\left(\boldsymbol{\mu}_k^{-}\right)-\mathbf{H}_k\left(\mathbf{x}_k-\boldsymbol{\mu}_k^{-}\right)\right)
\end{aligned}\tag{D.1}$$

Following Eq. (C.5), it can be easily shown that for a positive semi-definite quadratic form, there is inequality $\mathbf{x}^T\mathbf{A}\mathbf{x}=\sum\lambda_i\langle\mathbf{v}_i,\mathbf{x}\rangle^2\le\lambda_{\max}\|\mathbf{x}\|_2^{\,2}$, where $\mathbf{v}$ and λ are the singular vectors and values of matrix $\mathbf{A}$. In addition, since the Jacobian matrix of $\mathbf{g}$ is bounded $\|\mathbf{H}_k\|_2\le M_H$, based on the mean value theorem of vector-valued functions [219], the function $\mathbf{g}$ satisfies the Lipschitz condition $\|\mathbf{g}(\mathbf{x}_k)-\mathbf{g}(\boldsymbol{\mu}_k)\|\le M_H\|\mathbf{x}_k-\boldsymbol{\mu}_k\|$. Therefore, assume that the largest singular value of the noise covariance matrix $\mathbf{R}_k$ is λ_R, it can be shown that for the terms in Eq. (D.1):

$$G_1\le\lambda_R\|\mathbf{H}_k\|_2^{\,2}\left\|\left(\mathbf{x}_k-\boldsymbol{\mu}_k^{-}\right)\right\|_2^{\,2}\le\lambda_R M_H^{\,2}\left\|\mathbf{x}_k-\boldsymbol{\mu}_k^{-}\right\|_2^{\,2}\tag{D.2}$$

$$G_2\le\lambda_R\left\|\mathbf{g}(\mathbf{x}_k)-\mathbf{g}\left(\boldsymbol{\mu}_k^{-}\right)\right\|_2^{\,2}\le\lambda_R M_H^{\,2}\left\|\mathbf{x}_k-\boldsymbol{\mu}_k^{-}\right\|_2^{\,2}\tag{D.3}$$

$$|G_3|\le\lambda_R\left\|\left(\mathbf{y}_k-\mathbf{g}(\boldsymbol{\mu}_k)\right)\right\|_2\left\|\mathbf{r}_2(\mathbf{x}_k)\right\|_2\le\lambda_R M_g M_r\left\|\mathbf{x}_k-\boldsymbol{\mu}_k^{-}\right\|_2^2\tag{D.4}$$

Given that $E\left[\left\|\left(\mathbf{x}_k-\boldsymbol{\mu}_k^{-}\right)\right\|_2^{\,2}\right]=\mathrm{tr}\left(\boldsymbol{\Sigma}_k^{-}\right)$, the likelihood divergence is upper bounded by:

$$\int_{-\infty}^{\infty}p^{-}(\mathbf{x}_k)\left|\log\frac{p(\mathbf{y}_k|\mathbf{x}_k)}{\hat{p}(\mathbf{y}_k|\mathbf{x}_k)}\right|d\mathbf{x}_k\le\frac{1}{2}E\left[|G_1|+|G_2|+2|G_3|\right]\le c\,\mathrm{tr}\left(\boldsymbol{\Sigma}_k^{-}\right)\tag{D.5}$$

where $c=\lambda_R\left(M_H^{\,2}+M_gM_r\right)$. Therefore, the likelihood divergence can be made lower than any positive threshold th_L provided that the trace $\mathrm{tr}(\boldsymbol{\Sigma}_k^{-})<\sigma_L=th_L/c$.

Next, substitute Eq. (3.3) into Eq. (3.11), the KL divergence between estimated posterior PDF $\hat{p}(\mathbf{x}_k)=\mathcal{N}(\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k)$ and true PDF $p(\mathbf{x}_k)$ is:

$$
\begin{aligned}
&D_{KL}\left(\hat{p}\left(\mathbf{x}_k\right)\|\, p\left(\mathbf{x}_k\right)\right) \\
&= \int_{-\infty}^{\infty} \hat{p}\left(\mathbf{x}_k\right) \log \frac{\hat{p}\left(\mathbf{y}_k \mid \mathbf{x}_k\right)}{p\left(\mathbf{y}_k \mid \mathbf{x}_k\right)} d\mathbf{x}_k + \log \frac{\int_{-\infty}^{\infty} p\left(\mathbf{y}_k \mid \mathbf{x}_k\right) p^{-}\left(\mathbf{x}_k\right) d\mathbf{x}_k}{\int_{-\infty}^{\infty} \hat{p}\left(\mathbf{y}_k \mid \mathbf{x}_k\right) p^{-}\left(\mathbf{x}_k\right) d\mathbf{x}_k}
\end{aligned} \tag{D.6}
$$

The absolute value of the first term can be decomposed as:

$$
\begin{aligned}
&\left| \int_{-\infty}^{\infty} \hat{p}\left(\mathbf{x}_k\right) \log \frac{\hat{p}\left(\mathbf{y}_k \mid \mathbf{x}_k\right)}{p\left(\mathbf{y}_k \mid \mathbf{x}_k\right)} d\mathbf{x}_k \right| \\
&\leq \int_{-\infty}^{\infty} \frac{\hat{p}\left(\mathbf{y}_k \mid \mathbf{x}_k\right) p^{-}\left(\mathbf{x}_k\right)}{\int_{-\infty}^{\infty} \hat{p}\left(\mathbf{y}_k \mid \mathbf{x}_k\right) p^{-}\left(\mathbf{x}_k\right) d\mathbf{x}_k} \left| \log \frac{\hat{p}\left(\mathbf{y}_k \mid \mathbf{x}_k\right)}{p\left(\mathbf{y}_k \mid \mathbf{x}_k\right)} \right| d\mathbf{x}_k \\
&\leq d \int_{-\infty}^{\infty} p^{-}\left(\mathbf{x}_k\right) \left| \log \frac{p\left(\mathbf{y}_k \mid \mathbf{x}_k\right)}{\hat{p}\left(\mathbf{y}_k \mid \mathbf{x}_k\right)} \right| d\mathbf{x}_k \leq cd \operatorname{tr}\left(\boldsymbol{\Sigma}_k^{-}\right)
\end{aligned} \tag{D.7}
$$

where $d = \dfrac{\max \hat{p}\left(\mathbf{y}_k \mid \mathbf{x}_k\right)}{\int_{-\infty}^{\infty} \hat{p}\left(\mathbf{y}_k \mid \mathbf{x}_k\right) p^{-}\left(\mathbf{x}_k\right) d\mathbf{x}_k} = \dfrac{\left|2\pi \mathbf{R}_k\right|^{-1/2}}{\mathcal{N}\left(\mathbf{y}_k \mid \mathbf{g}\left(\boldsymbol{\mu}_k^{-}\right), \mathbf{R}_k + \mathbf{H}_k \boldsymbol{\Sigma}_k^{-} \mathbf{H}_k^{T}\right)}$ is derived by combining the marginal likelihood of EKF (pp. 214, Ref [73]) and the fact that $\max \hat{p}\left(\mathbf{y}_k \mid \mathbf{x}_k\right) = \left|2\pi \mathbf{R}_k\right|^{-1/2}$ when $\mathbf{y}_k = \mathbf{g}\left(\boldsymbol{\mu}_k^{-}\right) + \mathbf{H}_k\left(\mathbf{x}_k - \boldsymbol{\mu}_k^{-}\right)$. Using $\log \dfrac{p\left(\mathbf{y}_k \mid \mathbf{x}_k\right)}{\hat{p}\left(\mathbf{y}_k \mid \mathbf{x}_k\right)} \leq c \left\| \mathbf{x}_k - \boldsymbol{\mu}_k^{-} \right\|_2^{\,2}$ from Eq. (D.5), the fraction term in Eq. (D.6) can be rewritten as:

$$
\begin{aligned}
&\frac{\int_{-\infty}^{\infty} p\left(\mathbf{y}_k \mid \mathbf{x}_k\right) p^{-}\left(\mathbf{x}_k\right) d\mathbf{x}_k}{\int_{-\infty}^{\infty} \hat{p}\left(\mathbf{y}_k \mid \mathbf{x}_k\right) p^{-}\left(\mathbf{x}_k\right) d\mathbf{x}_k} = \int_{-\infty}^{\infty} \frac{\hat{p}\left(\mathbf{y}_k \mid \mathbf{x}_k\right) p^{-}\left(\mathbf{x}_k\right)}{\int_{-\infty}^{\infty} \hat{p}\left(\mathbf{y}_k \mid \mathbf{x}_k\right) p^{-}\left(\mathbf{x}_k\right) d\mathbf{x}_k} \frac{p\left(\mathbf{y}_k \mid \mathbf{x}_k\right)}{\hat{p}\left(\mathbf{y}_k \mid \mathbf{x}_k\right)} d\mathbf{x}_k \\
&= E_{\hat{p}(\mathbf{x}_k)}\left[\frac{p\left(\mathbf{y}_k \mid \mathbf{x}_k\right)}{\hat{p}\left(\mathbf{y}_k \mid \mathbf{x}_k\right)} \right] \leq E_{\hat{p}(\mathbf{x}_k)}\left[\exp\left(c \left\| \mathbf{x}_k - \boldsymbol{\mu}_k^{-} \right\|_2^{\,2} \right) \right] \\
&= E_{\hat{p}(\mathbf{x}_k)}\left[\exp\left(-\frac{1}{2}\left(\mathbf{x}_k - \boldsymbol{\mu}_k^{-}\right)^{T} \left(-\frac{1}{2c} \mathbf{I}_n \right)^{-1} \left(\mathbf{x}_k - \boldsymbol{\mu}_k^{-}\right) \right) \right] \\
&= \frac{1}{\sqrt{\left|\mathbf{I}_n - 2c\boldsymbol{\Sigma}_k\right|}} \exp\left(-\frac{1}{2}\left(\boldsymbol{\mu}_k - \boldsymbol{\mu}_k^{-}\right)^{T} \left(\boldsymbol{\Sigma}_k - \frac{1}{2c} \mathbf{I}_n \right)^{-1} \left(\boldsymbol{\mu}_k - \boldsymbol{\mu}_k^{-}\right) \right)
\end{aligned} \tag{D.8}
$$

where the expectation integral is derived using the formula for the product of two Gaussians (Results 8.2, pp. 169 in [220]). Taking the natural logarithm of Eq. (D.8) gives:

$$\log E_{\hat{p}(\mathbf{x}_k)}\left[\frac{p(\mathbf{y}_k \mid \mathbf{x}_k)}{\hat{p}(\mathbf{y}_k \mid \mathbf{x}_k)}\right] \le -\frac{1}{2}\log\left|\mathbf{I}_n - 2c\boldsymbol{\Sigma}_k\right| + c\left(\boldsymbol{\mu}_k - \boldsymbol{\mu}_k^-\right)^T \left(\mathbf{I}_n - 2c\boldsymbol{\Sigma}_k\right)^{-1}\left(\boldsymbol{\mu}_k - \boldsymbol{\mu}_k^-\right) \quad \text{(D.9)}$$

In the EKF solution [73], $\boldsymbol{\mu}_k - \boldsymbol{\mu}_k^- = \mathbf{K}\left(\mathbf{y}_k - \mathbf{g}\left(\boldsymbol{\mu}_k^-\right)\right)$ with $\mathbf{K} = \boldsymbol{\Sigma}_k^- \mathbf{H}_k \boldsymbol{\Sigma}_{yy}^{-1}$. Hence, the second term is in the order of $\|\boldsymbol{\Sigma}_k^-\|^2$. Using Eq. (1) in [221], it has $-\log\left|\mathbf{I}_n - 2c\boldsymbol{\Sigma}_k\right| = 2c\,\mathrm{tr}\left(\boldsymbol{\Sigma}_k\right) + r_2$, where r_2 is a second-order remainder. Also, $\mathrm{tr}\left(\boldsymbol{\Sigma}_k\right) \le \mathrm{tr}\left(\boldsymbol{\Sigma}_k^-\right)$ given that $\boldsymbol{\Sigma}_k = \boldsymbol{\Sigma}_k^- - \mathbf{K}\boldsymbol{\Sigma}_{yy}\mathbf{K}^T$. If $\mathrm{tr}(\boldsymbol{\Sigma}_k^-) \ll 1$, $\mathrm{tr}(\boldsymbol{\Sigma}_k^-)^2$ and $\|\boldsymbol{\Sigma}_k^-\|^2$ will be less than $\mathrm{tr}(\boldsymbol{\Sigma}_k^-)$ by orders of magnitude. Therefore, the KL divergence bounded by the sum of Eq. (D.7) and (D.9) will decay to zero with a linear rate of $\mathrm{tr}(\boldsymbol{\Sigma}_k^-)$:

$$D_{KL}\left(\hat{p}(\mathbf{x}_k) \,\|\, p(\mathbf{x}_k)\right) \approx c(d+1)\,\mathrm{tr}\left(\boldsymbol{\Sigma}_k^-\right) + H.O.T \le cM\,\mathrm{tr}\left(\boldsymbol{\Sigma}_k^-\right) \quad \text{(D.10)}$$

where *H.O.T* denotes the higher-order terms of $\mathrm{tr}(\boldsymbol{\Sigma}_k^-)^2$ or $\|\boldsymbol{\Sigma}_k^-\|^2$, and M is a finite constant $d+1 < M < \infty$. Given that $\mathrm{tr}(\boldsymbol{\Sigma}_k^-) < th_L/c$, the KL divergence will be less than $M \times th_L$. Q.E.D.

REFERENCES

[1] Dhar, V., 2013. Data science and prediction. Communications of the ACM, 56(12), pp.64-73.

[2] Chen, J., Tao, Y., Wang, H. and Chen, T., 2015. Big data based fraud risk management at Alibaba. The Journal of Finance and Data Science, 1(1), pp.1-10.

[3] Hiransha, M., Gopalakrishnan, E.A., Menon, V.K. and Soman, K.P., 2018. NSE stock market prediction using deep-learning models. Procedia computer science, 132, pp.1351-1362.

[4] Singh, S., Pandey, S.K., Pawar, U. and Janghel, R.R., 2018. Classification of ECG arrhythmia using recurrent neural networks. Procedia computer science, 132, pp.1290-1297.

[5] Razzak, M.I., Naz, S. and Zaib, A., 2018. Deep learning for medical image processing: Overview, challenges and the future. In Classification in BioApps (pp. 323-350). Springer, Cham.

[6] Hou, Z.S. and Wang, Z., 2013. From model-based control to data-driven control: Survey, classification and perspective. Information Sciences, 235, pp.3-35.

[7] Stetco, A., Dinmohammadi, F., Zhao, X., Robu, V., Flynn, D., Barnes, M., Keane, J. and Nenadic, G., 2019. Machine learning methods for wind turbine condition monitoring: A review. Renewable energy, 133, pp.620-635.

[8] Diez-Olivan, A., Del Ser, J., Galar, D. and Sierra, B., 2019. Data fusion and machine learning for industrial prognosis: Trends and perspectives towards Industry 4.0. Information Fusion, 50, pp.92-111.

[9] Yin, S., Li, X., Gao, H. and Kaynak, O., 2014. Data-based techniques focused on modern industry: An overview. IEEE Transactions on Industrial Electronics, 62(1), pp.657-667.

[10] Du, J. and Xu, Y., 2017. Hierarchical deep neural network for multivariate regression. Pattern Recognition, 63, pp.149-157.

[11] Basu, J.K., Bhattacharyya, D. and Kim, T.H., 2010. Use of artificial neural network in pattern recognition. International journal of software engineering and its applications, 4(2).

[12] Schmidhuber, J., 2015. Deep learning in neural networks: An overview. Neural networks, 61, pp.85-117.

[13] Anghinoni, L., Zhao, L., Ji, D. and Pan, H., 2019. Time series trend detection and forecasting using complex network topology analysis. Neural Networks, 117, pp.295-306.

[14] Kendall, A. and Gal, Y., 2017. What uncertainties do we need in Bayesian deep learning for computer vision?. Advances in neural information processing systems, pp.5574-5584.

[15] Gal, Y., 2016. Uncertainty in deep learning. Doctoral dissertation, University of Cambridge.

[16] Blundell, C., Cornebise, J., Kavukcuoglu, K. and Wierstra, D., 2015, June. Weight uncertainty in neural network. In International Conference on Machine Learning (pp. 1613-1622). PMLR.

[17] Ngo, P.D. and Shin, Y.C., 2016. Modeling of unstructured uncertainties and robust controlling of nonlinear dynamic systems based on type-2 fuzzy basis function networks. Engineering Applications of Artificial Intelligence, 53, pp.74-85.

[18] Xu, L., Wang, J. and Chen, Q., 2012. Kalman filtering state of charge estimation for battery management system based on a stochastic fuzzy neural network battery model. Energy Conversion and Management, 53(1), pp.33-39.

[19] Buehler, E., 2017. Efficient Uncertainty Propagation for Stochastic Model Predictive Control. Doctoral dissertation, UC Berkeley, Berkeley, USA.

[20] Venkatasubramanian, V., Rengaswamy, R., Yin, K. and Kavuri, S.N., 2003. A review of process fault detection and diagnosis: Part I: Quantitative model-based methods. Computers & chemical engineering, 27(3), pp.293-311.

[21] Tidriri, K., Chatti, N., Verron, S. and Tiplica, T., 2016. Bridging data-driven and model-based approaches for process fault diagnosis and health monitoring: A review of researches and future challenges. Annual Reviews in Control, 42, pp.63-81.

[22] Buswell, R.A., 2001. Uncertainty in the first principle model based condition monitoring of HVAC systems (Doctoral dissertation, Loughborough University).

[23] Rigatos, G. and Siano, P., 2016. Power transformers' condition monitoring using neural modeling and the local statistical approach to fault diagnosis. International Journal of Electrical Power & Energy Systems, 80, pp.150-159.

[24] Singh, G.K. and Ahmed, S.A.K.S.A., 2004. Vibration signal analysis using wavelet transform for isolation and identification of electrical faults in induction machine. Electric Power Systems Research, 68(2), pp.119-136.

[25] Tian, Z., 2012. An artificial neural network method for remaining useful life prediction of equipment subject to condition monitoring. Journal of Intelligent Manufacturing, 23(2), pp.227-237.

[26] Nilsson, J. and Bertling, L., 2007. Maintenance management of wind power systems using condition monitoring systems—life cycle cost analysis for two case studies. IEEE Transactions on energy conversion, 22(1), pp.223-229.

[27] Huynh, K.T., Barros, A. and Bérenguer, C., 2012. Maintenance decision-making for systems operating under indirect condition monitoring: value of online information and impact of measurement uncertainty. IEEE Transactions on Reliability, 61(2), pp.410-425.

[28] Ettler, P. and Dedecius, K., 2014, September. Quantification of information uncertainty for the purpose of condition monitoring. In 2014 11th International Conference on Informatics

in Control, Automation and Robotics (ICINCO) (Vol. 1, pp. 127-132). IEEE.

[29] Aizpurua, J.I., Stewart, B.G., McArthur, S.D.J., Lambert, B., Cross, J.G. and Catterson, V.M., 2019. Improved power transformer condition monitoring under uncertainty through soft computing and probabilistic health index. Applied Soft Computing, 85, p.105530.

[30] DebRoy, T., Wei, H.L., Zuback, J.S., Mukherjee, T., Elmer, J.W., Milewski, J.O., Beese, A.M., Wilson-Heid, A., De, A. and Zhang, W., 2017. Additive manufacturing of metallic components–process, structure and properties. Progress in Materials Science.

[31] Grasso, M. and Colosimo, B.M., 2017. Process defects and in situ monitoring methods in metal powder bed fusion: a review. Measurement Science and Technology, 28(4), p.044005.

[32] Shamsaei, N., Yadollahi, A., Bian, L. and Thompson, S.M., 2015. An overview of Direct Laser Deposition for additive manufacturing; Part II: Mechanical behavior, process parameter optimization and control. Additive Manufacturing, 8, pp.12-35.

[33] PwC, 3D Printing and the New Shape of Industrial Manufacturing, Price Waterhouse Coopers LLP, Delaware, 2014

[34] Energetics Incorporated, Measurement Science Roadmap for Metal-Based Additive Manufacturing, National Institute of Standards and Technology, Maryland, US., 2013

[35] Tapia, G. and Elwany, A., 2014. A review on process monitoring and control in metal-based additive manufacturing. Journal of Manufacturing Science and Engineering, 136(6), p.060801.

[36] Everton, S.K., Hirsch, M., Stravroulakis, P., Leach, R.K. and Clare, A.T., 2016. Review of in-situ process monitoring and in-situ metrology for metal additive manufacturing. Materials & Design, 95, pp.431-445.

[37] O'Donnell, G., Young, P., Kelly, K. and Byrne, G., 2001. Towards the improvement of tool condition monitoring systems in the manufacturing environment. Journal of Materials Processing Technology, 119(1-3), pp.133-139.

[38] de Jesús Rubio, J., 2017. Stable Kalman filter and neural network for the chaotic systems identification. Journal of the Franklin Institute, 354(16), pp.7444-7462.

[39] Chen, M., Ge, S.S. and How, B.V.E., 2010. Robust adaptive neural network control for a class of uncertain MIMO nonlinear systems with input nonlinearities. IEEE Transactions on Neural Networks, 21(5), pp.796-812.

[40] He, W., Chen, Y. and Yin, Z., 2015. Adaptive neural network control of an uncertain robot with full-state constraints. IEEE transactions on cybernetics, 46(3), pp.620-629.

[41]

[42] Kabir, H.D., Khosravi, A., Hosen, M.A. and Nahavandi, S., 2018. Neural network-based uncertainty quantification: A survey of methodologies and applications. IEEE access, 6, pp.36218-36234.

[43] Abdar, M., Pourpanah, F., Hussain, S., Rezazadegan, D., Liu, L., Ghavamzadeh, M., Fieguth, P., Cao, X., Khosravi, A., Acharya, U.R. and Makarenkov, V., 2021. A review of uncertainty quantification in deep learning: Techniques, applications and challenges. Information Fusion.

[44] Chen, C., Carlson, D., Gan, Z., Li, C. and Carin, L., 2016, May. Bridging the gap between stochastic gradient MCMC and stochastic optimization. In Artificial Intelligence and Statistics (pp. 1051-1060). PMLR.

[45] Huber, M.F., 2011, July. Adaptive Gaussian mixture filter based on statistical linearization. In 14th International Conference on Information Fusion (pp. 1-8). IEEE.

[46] Gast, J. and Roth, S., 2018. Lightweight probabilistic deep networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 3369-3378).

[47] Frey, B.J. and Hinton, G.E., 1999. Variational learning in nonlinear Gaussian belief networks. Neural Computation, 11(1), pp.193-213.

[48] Wang, H., Shi, X. and Yeung, D.Y., 2016. Natural-parameter networks: A class of probabilistic neural networks. Advances in Neural Information Processing Systems, 29, pp.118-126.

[49] Abdelaziz, A.H., Watanabe, S., Hershey, J.R., Vincent, E. and Kolossa, D., 2015. Uncertainty Propagation Through Deep Neural Networks. In Sixteenth Annual Conference of the International Speech Communication Association. Dresden, Germany.

[50] Bui, T., Hernández-Lobato, D., Hernandez-Lobato, J., Li, Y. and Turner, R., 2016, June. Deep Gaussian processes for regression using approximate expectation propagation. In International conference on machine learning (pp. 1472-1481).

[51] Lee, J., Bahri, Y., Novak, R., Schoenholz, S.S., Pennington, J. and Sohl-Dickstein, J., 2017. Deep neural networks as gaussian processes. arXiv preprint arXiv:1711.00165.

[52] Julier, S.J. and Uhlmann, J.K., 2004. Unscented filtering and nonlinear estimation. Proceedings of the IEEE, 92(3), pp.401-422.

[53] Dong, G., Zhang, X., Zhang, C. and Chen, Z., 2015. A method for state of energy estimation of lithium-ion batteries based on neural network model. Energy, 90, pp.879-888.

[54] Vlassis, N. and Likas, A., 2002. A greedy EM algorithm for Gaussian mixture learning. Neural processing letters, 15(1), pp.77-87.

[55] Psiaki, M.L., Schoenberg, J.R. and Miller, I.T., 2015. Gaussian sum reapproximation for use in a nonlinear filter. Journal of Guidance, Control, and Dynamics, 38(2), pp.292-303.

[56]

[57] Terejanu, G., Singla, P., Singh, T. and Scott, P.D., 2008. Uncertainty propagation for nonlinear dynamic systems using Gaussian mixture models. Journal of Guidance, Control, and Dynamics, 31(6), pp.1623-1633.

[58] Terejanu, G., Singla, P., Singh, T. and Scott, P.D., 2011. Adaptive Gaussian sum filter for

nonlinear Bayesian estimation. IEEE Transactions on Automatic Control, 56(9), pp.2151-2156.

[59] Faubel, F., McDonough, J. and Klakow, D., 2009. The split and merge unscented Gaussian mixture filter. IEEE Signal Processing Letters, 16(9), pp.786-789.

[60] Horwood, J.T. and Poore, A.B., 2011. Adaptive Gaussian sum filters for space surveillance. IEEE transactions on automatic control, 56(8), pp.1777-1790.

[61] DeMars, K.J., Bishop, R.H. and Jah, M.K., 2013. Entropy-based approach for uncertainty propagation of nonlinear dynamical systems. Journal of Guidance, Control, and Dynamics, 36(4), pp.1047-1057.

[62] Tuggle, K. and Zanetti, R., 2018. Automated Splitting Gaussian Mixture Nonlinear Measurement Update. Journal of Guidance, Control, and Dynamics, 41(3), pp.725-734.

[63] Auger, F., Hilairet, M., Guerrero, J.M., Monmasson, E., Orlowska-Kowalska, T. and Katsura, S., 2013. Industrial applications of the Kalman filter: A review. IEEE Transactions on Industrial Electronics, 60(12), pp.5458-5471.

[64] Chen, Z., 2003. Bayesian filtering: From Kalman filters to particle filters, and beyond. Statistics, 182(1), pp.1-69.

[65] Kalman, R.E., 1960. A new approach to linear filtering and prediction problems.

[66] Wan, E.A. and Van Der Merwe, R., 2000, October. The unscented Kalman filter for nonlinear estimation. In Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No. 00EX373) (pp. 153-158). IEEE.

[67] Ito, K. and Xiong, K., 2000. Gaussian filters for nonlinear filtering problems. IEEE transactions on automatic control, 45(5), pp.910-927.

[68] Arasaratnam, I. and Haykin, S., 2009. Cubature kalman filters. IEEE Transactions on automatic control, 54(6), pp.1254-1269.

[69] Evensen, G., 2003. The ensemble Kalman filter: Theoretical formulation and practical implementation. Ocean dynamics, 53(4), pp.343-367.

[70] Gordon, N.J., Salmond, D.J. and Smith, A.F., 1993, April. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. In IEE Proceedings F-radar and signal processing (Vol. 140, No. 2, pp. 107-113). IET.

[71] Arulampalam, M.S., Maskell, S., Gordon, N. and Clapp, T., 2002. A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. IEEE Transactions on signal processing, 50(2), pp.174-188.

[72] Beskos, A., Crisan, D., Jasra, A., Kamatani, K. and Zhou, Y., 2017. A stable particle filter for a class of high-dimensional state-space models. Advances in Applied Probability, 49(1), pp.24-48.

[73] Anderson, B.D. and Moore, J.B., 2012. Optimal filtering. Courier Corporation.

[74] Sorenson, H.W. and Alspach, D.L., 1971. Recursive Bayesian estimation using Gaussian sums. Automatica, 7(4), pp.465-479.

[75] Arasaratnam, I., Haykin, S. and Elliott, R.J., 2007. Discrete-time nonlinear filtering algorithms using Gauss–Hermite quadrature. Proceedings of the IEEE, 95(5), pp.953-977.

[76] Tam, W.I., Plataniotis, K.N. and Hatzinakos, D., 1999. An adaptive Gaussian sum algorithm for radar tracking. Signal processing, 77(1), pp.85-104.

[77] Kotecha, J.H. and Djuric, P.M., 2003. Gaussian sum particle filtering. IEEE Transactions on signal processing, 51(10), pp.2602-2612.

[78] Raihan, D. and Chakravorty, S., 2018. Particle Gaussian mixture filters-I. Automatica, 98, pp.331-340.

[79] Stordal, A.S., Karlsen, H.A., Nævdal, G., Skaug, H.J. and Vallès, B., 2011. Bridging the ensemble Kalman filter and particle filters: the adaptive Gaussian mixture filter. Computational Geosciences, 15(2), pp.293-305.

[80] Psiaki, M.L., 2016. Gaussian mixture nonlinear filtering with resampling for mixand narrowing. IEEE Transactions on Signal Processing, 64(21), pp.5499-5512.

[81] Leong, P.H., Arulampalam, S., Lamahewa, T.A. and Abhayapala, T.D., 2013. A Gaussian-sum based cubature Kalman filter for bearings-only tracking. IEEE Transactions on Aerospace and Electronic Systems, 49(2), pp.1161-1176.

[82] Khosravi, A., Nahavandi, S., Creighton, D. and Atiya, A.F., 2011. Comprehensive review of neural network-based prediction intervals and new advances. IEEE Transactions on neural networks, 22(9), pp.1341-1356.

[83] Juang, C.F., Huang, R.B. and Cheng, W.Y., 2010. An interval type-2 fuzzy-neural network with support-vector regression for noisy regression problems. IEEE Transactions on fuzzy systems, 18(4), pp.686-699.

[84] Sadeghi, J., De Angelis, M. and Patelli, E., 2019. Efficient training of interval Neural Networks for imprecise training data. Neural Networks, 118, pp.338-351

[85] Khosravi, A., Nahavandi, S., Srinivasan, D. and Khosravi, R., 2014. Constructing optimal prediction intervals by using neural networks and bootstrap method. IEEE transactions on neural networks and learning systems, 26(8), pp.1810-1815.

[86] Cheng, L., Zang, H., Ding, T., Sun, R., Wang, M., Wei, Z. and Sun, G., 2018. Ensemble recurrent neural network based probabilistic wind speed forecasting approach. Energies, 11(8), p.1958.

[87] Jospin, L.V., Buntine, W., Boussaid, F., Laga, H. and Bennamoun, M., 2020. Hands-on Bayesian Neural Networks-a Tutorial for Deep Learning Users. arXiv preprint arXiv:2007.06823.

[88] Neal, R.M., 1995. Bayesian Learning for Neural Networks. Doctoral dissertation,

University of Toronto.

[89] Deng, W., Zhang, X., Liang, F. and Lin, G., 2019. An adaptive empirical Bayesian method for sparse deep learning. In Advances in neural information processing systems, (p.5563).

[90] Hernández-Lobato, J.M. and Adams, R., 2015, June. Probabilistic backpropagation for scalable learning of bayesian neural networks. In International Conference on Machine Learning (pp. 1861-1869).

[91] Graves, A., 2011. Practical variational inference for neural networks. In Advances in neural information processing systems, (pp.2348-2356).

[92] Gal, Y. and Ghahramani, Z., 2016, June. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In international conference on machine learning (pp. 1050-1059).

[93] Gal, Y., Hron, J. and Kendall, A., 2017. Concrete dropout. arXiv preprint arXiv:1705.07832.

[94] Hernandez-Lobato, J., Li, Y., Rowland, M., Bui, T., Hernández-Lobato, D. and Turner, R., 2016, June. Black-box alpha divergence minimization. In International Conference on Machine Learning (pp. 1511-1520).

[95] Zhao, J., Liu, X., He, S. and Sun, S., 2020. Probabilistic inference of Bayesian neural networks with generalized expectation propagation. Neurocomputing, 412, pp.392-398.

[96] Jylänki, P., Nummenmaa, A. and Vehtari, A., 2014. Expectation propagation for neural networks with sparsity-promoting priors. The Journal of Machine Learning Research, 15(1), pp.1849-1901.

[97] Tran, D., Dusenberry, M., van der Wilk, M. and Hafner, D., 2019. Bayesian layers: A module for neural network uncertainty. In Advances in Neural Information Processing Systems (pp.14633-14645).

[98] Pawlowski, N., Brock, A., Lee, M.C., Rajchl, M. and Glocker, B., 2017. Implicit weight uncertainty in neural networks. arXiv preprint arXiv:1711.01297.

[99] Bishop, C.M., 1994. Mixture density networks (p. 7). Technical Report NCRG/4288, Aston University, Birmingham, UK.

[100] Zen, H. and Senior, A., 2014, May. Deep mixture density networks for acoustic modeling in statistical parametric speech synthesis. In Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on (pp. 3844-3848). IEEE.

[101] Zhang, J., Yan, J., Infield, D., Liu, Y. and Lien, F.S., 2019. Short-term forecasting and uncertainty analysis of wind turbine power based on long short-term memory network and Gaussian mixture model. Applied energy, 241, pp.229-244.

[102] Krizhevsky, A., Sutskever, I. and Hinton, G.E., 2012. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems (pp.

1097-1105).

[103] Simonyan, K. and Zisserman, A., 2014. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

[104] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. and Rabinovich, A., 2015. Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1-9).

[105] He, K., Zhang, X., Ren, S. and Sun, J., 2016. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

[106] Iandola, F.N., Han, S., Moskewicz, M.W., Ashraf, K., Dally, W.J. and Keutzer, K., 2016. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size. arXiv preprint arXiv:1602.07360.

[107] Szegedy, C., Ioffe, S., Vanhoucke, V. and Alemi, A.A., 2017, February. Inception-v4, inception-resnet and the impact of residual connections on learning. In AAAI (Vol. 4, p. 12).

[108] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M. and Berg, A.C., 2015. Imagenet large scale visual recognition challenge. International Journal of Computer Vision, 115(3), pp.211-252.

[109] Alom, M.Z., Taha, T.M., Yakopcic, C., Westberg, S., Hasan, M., Van Esesn, B.C., Awwal, A.A.S. and Asari, V.K., 2018. The History Began from AlexNet: A Comprehensive Survey on Deep Learning Approaches. arXiv preprint arXiv:1803.01164.

[110] MathWorks, 2018. Pretrained Convolutional Neural Networks, Matlab ver. 2018b, https: //www.mathworks.com/help/deeplearning/ug/pretrained-convolutional-neural-networks.html

[111] Teti, R., Jemielniak, K., O'Donnell, G., and Dornfeld, D., 2010. Advanced Monitoring of Machining Operations, CIRP Annals-Manufacturing Technology, 59(2), pp. 717-739.

[112] Abellan-Nebot, J. V. and Subirón, F. R., 2010. A Review of Machining Monitoring Systems Based on Artificial Intelligence Process Models. The International Journal of Advanced Manufacturing Technology, 47(1-4), pp. 237-257.

[113] Sick, B., 2002. On-Line and Indirect Tool Wear Monitoring in Turning with Artificial Neural Networks: A Review of More Than a Decade of Research. Mechanical Systems and Signal Processing, 16(4), pp. 487-546.

[114] Grzesik, W., 2008. Influence of Tool Wear on Surface Roughness in Hard Turning Using Differently Shaped Ceramic Tools. Wear, 265(3-4), pp. 327-335.

[115] Niaki, F.A. and Mears, L., 2017. A comprehensive study on the effects of tool wear on surface roughness, dimensional integrity and residual stress in turning IN718 hard-to-machine alloy. Journal of Manufacturing Processes, 30, pp.268-280.

[116] Liu, T.-I. and Jolley, B., 2015. Tool Condition Monitoring (TCM) Using Neural Networks. The International Journal of Advanced Manufacturing Technology, 78(9-12), pp. 1999-2007.

[117] Nouri, M., Fussell, B. K., Ziniti, B. L., and Linder, E., 2015. Real-Time Tool Wear Monitoring in Milling Using a Cutting Condition Independent Method. International Journal of Machine Tools and Manufacture, 89, pp. 1-13.

[118] Li, N., Chen, Y., Kong, D., and Tan, S., 2017. Force-Based Tool Condition Monitoring for Turning Process Using V-Support Vector Regression. The International Journal of Advanced Manufacturing Technology, 91(1-4), pp. 351-361.

[119] Scheffer, C. and Heyns, P., 2001. Wear Monitoring in Turning Operations Using Vibration and Strain Measurements. Mechanical systems and signal processing, 15(6), pp. 1185-1202.

[120] Dimla, D. E., 2002. The Correlation of Vibration Signal Features to Cutting Tool Wear in a Metal Turning Operation. The International Journal of Advanced Manufacturing Technology, 19(10), pp. 705-713.

[121] Alonso, F. and Salgado, D., 2008. Analysis of the Structure of Vibration Signals for Tool Wear Detection. Mechanical Systems and Signal Processing, 22(3), pp. 735-748.

[122] Prasad, B. S. and Babu, M. P., 2017. Correlation between Vibration Amplitude and Tool Wear in Turning: Numerical and Experimental Analysis. Engineering Science and Technology, an International Journal, 20(1), pp. 197-211.

[123] Li, X., 2002. A Brief Review: Acoustic Emission Method for Tool Wear Monitoring During Turning. International Journal of Machine Tools and Manufacture, 42(2), pp. 157-165.

[124] Ren, Q., Balazinski, M., Baron, L., Jemielniak, K., Botez, R., and Achiche, S., 2014. Type-2 Fuzzy Tool Condition Monitoring System Based on Acoustic Emission in Micromilling. Information Sciences, 255, pp. 121-134.

[125] Maia, L. H. A., Abrao, A. M., Vasconcelos, W. L., Sales, W. F., and Machado, A. R., 2015. A New Approach for Detection of Wear Mechanisms and Determination of Tool Life in Turning Using Acoustic Emission. Tribology International, 92, pp. 519-532.

[126] Axinte, D. and Gindy, N., 2004. Assessment of the Effectiveness of a Spindle Power Signal for Tool Condition Monitoring in Machining Processes. International journal of production research, 42(13), pp. 2679-2691.

[127] Drouillet, C., Karandikar, J., Nath, C., Journeaux, A.-C., El Mansori, M., and Kurfess, T., 2016. Tool Life Predictions in Milling Using Spindle Power with the Neural Network Technique. Journal of Manufacturing Processes, 22, pp. 161-168.

[128] Zhu, K., San Wong, Y., and Hong, G. S., 2009. Wavelet Analysis of Sensor Signals for Tool Condition Monitoring: A Review and Some New Results. International Journal of Machine Tools and Manufacture, 49(7), pp. 537-553.

[129] Niaki, F.A., Feng, L., Ulutan, D. and Mears, L., 2016. A Wavelet-based Data-driven Modelling for Tool Wear Assessment of Difficult to Machine Materials. International Journal of Mechatronics and Manufacturing Systems, 9(2), pp.97-121.

[130] Segreto, T., Simeone, A., and Teti, R., 2013. Multiple Sensor Monitoring in Nickel Alloy Turning for Tool Wear Assessment Via Sensor Fusion. Procedia CIRP, 12, pp. 85-90.

[131] Wang, G., Zhang, Y., Liu, C., Xie, Q. and Xu, Y., 2019. A new tool wear monitoring method based on multi-scale PCA. Journal of Intelligent Manufacturing, 30(1), pp.113-122.

[132] Guyon, I. and Elisseeff, A., 2003. An Introduction to Variable and Feature Selection. Journal of machine learning research, 3(Mar), pp. 1157-1182.

[133] Liao, T. W., 2010. Feature Extraction and Selection from Acoustic Emission Signals with an Application in Grinding Wheel Condition Monitoring. Engineering Applications of Artificial Intelligence, 23(1), pp. 74-84.

[134] Subrahmanya, N. and Shin, Y. C., 2008. Automated Sensor Selection and Fusion for Monitoring and Diagnostics of Plunge Grinding. Journal of manufacturing science and engineering, 130(3), pp. 031014.

[135] Wang, G. and Cui, Y., 2013. On Line Tool Wear Monitoring Based on Auto Associative Neural Network. Journal of Intelligent Manufacturing, 24(6), pp. 1085-1094.

[136] D'Addona, D.M., Ullah, A.S. and Matarazzo, D., 2017. Tool-wear Prediction and Pattern-recognition Using Artificial Neural Network and DNA-based Computing. Journal of Intelligent Manufacturing, 28(6), pp. 1285-1301.

[137] Aliustaoglu, C., Ertunc, H.M. and Ocak, H., 2009. Tool wear condition monitoring using a sensor fusion model based on fuzzy inference system. Mechanical Systems and Signal Processing, 23(2), pp.539-546.

[138] Gajate, A., Haber, R., Del Toro, R., Vega, P., and Bustillo, A., 2012. Tool Wear Monitoring Using Neuro-Fuzzy Techniques: A Comparative Study in a Turning Process. Journal of Intelligent Manufacturing, 23(3), pp. 869-882.

[139] Rizal, M., Ghani, J.A., Nuawi, M.Z. and Haron, C.H.C., 2013. Online tool wear prediction system in the turning process using an adaptive neuro-fuzzy inference system. Applied Soft Computing, 13(4), pp.1960-1968.

[140] Shi, D. and Gindy, N. N., 2007. Tool Wear Predictive Model Based on Least Squares Support Vector Machines. Mechanical Systems and Signal Processing, 21(4), pp. 1799-1814.

[141] Mehrabi, M. G. and Kannatey-Asibu Jr, E., 2002. Hidden Markov Model-Based Tool Wear Monitoring in Turning. Journal of Manufacturing Science and Engineering, 124(3), pp. 651-658.

[142] Yu, J., Liang, S., Tang, D., and Liu, H., 2017. A Weighted Hidden Markov Model Approach for Continuous-State Tool Wear Monitoring and Tool Life Prediction. The International Journal of Advanced Manufacturing Technology, 91(1-4), pp. 201-211.

[143] Wang, G., Qian, L., and Guo, Z., 2013. Continuous Tool Wear Prediction Based on Gaussian Mixture Regression Model. The International Journal of Advanced Manufacturing Technology, pp. 1-9.

[144] Wu, D., Jennings, C., Terpenny, J., Gao, R. X., and Kumara, S., 2017. A Comparative Study on Machine Learning Algorithms for Smart Manufacturing: Tool Wear Prediction Using Random Forests. Journal of Manufacturing Science and Engineering, 139(7), pp. 071018.

[145] Chen, Y., Jin, Y. and Jiri, G., 2018. Predicting tool wear with multi-sensor data using deep belief networks. The International Journal of Advanced Manufacturing Technology, 99(5-8), pp.1917-1926.

[146] Karandikar, J.M., Abbas, A.E. and Schmitz, T.L., 2014. Tool Life Prediction Using Bayesian Updating. Part 1: Milling Tool Life Model Using a Discrete Grid Method. Precision Engineering, 38(1), pp. 9-17.

[147] Karandikar, J.M., Abbas, A.E. and Schmitz, T.L., 2014. Tool Life Prediction Using Bayesian Updating. Part 2: Turning Tool Life Using a Markov Chain Monte Carlo Approach. Precision Engineering, 38(1), pp. 18-27.

[148] Akhavan Niaki, F., Ulutan, D. and Mears, L., 2016. Parameter inference under uncertainty in end-milling γ′-strengthened difficult-to-machine alloy. Journal of Manufacturing Science and Engineering, 138(6).

[149] Ren, Q., Balazinski, M., and Baron, L., 2009. Uncertainty Prediction for Tool Wear Condition Using Type-2 Tsk Fuzzy Approach. Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics, pp. 660-665.

[150] Niaki, F.A., Michel, M. and Mears, L., 2016. State of Health Monitoring in Machining: Extended Kalman Filter for Tool Wear Assessment in Turning of IN718 Hard-to-machine Alloy. Journal of Manufacturing Processes, 24, pp. 361-369.

[151] Wang, J., Wang, P. and Gao, R.X., 2015. Enhanced Particle Filter for Tool Wear Prediction. Journal of Manufacturing Systems, 36, pp. 35-45.

[152] Zhang, J., Starly, B., Cai, Y., Cohen, P.H. and Lee, Y.S., 2017. Particle Learning in Online Tool Wear Diagnosis and Prognosis. Journal of Manufacturing Processes, 28, pp. 457-463.

[153] Niaki, F.A., Ulutan, D. and Mears, L., 2015. Stochastic tool wear assessment in milling difficult to machine alloys. International Journal of Mechatronics and Manufacturing Systems, 8(3-4), pp.134-159.

[154] Akhavan Niaki, F. and Mears, L., 2018. A probabilistic-based study on fused direct and

indirect methods for tracking tool flank wear of Rene-108, nickel-based alloy. Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture, 232(11), pp.2030-2043.

[155] Kong, D., Chen, Y., Li, N., Duan, C., Lu, L. and Chen, D., 2019. Relevance vector machine for tool wear prediction. Mechanical Systems and Signal Processing, 127, pp.573-594.

[156] Gouarir, A., Martínez-Arellano, G., Terrazas, G., Benardos, P. and Ratchev, S., 2018. In-process tool wear prediction system based on machine learning techniques and force analysis. Procedia CIRP, 77, pp.501-504.

[157] Zheng, H. and Lin, J., 2019, June. A Deep Learning Approach for High Speed Machining Tool Wear Monitoring. In 2019 3rd International Conference on Robotics and Automation Sciences (ICRAS) (pp. 63-68). IEEE.

[158] Martínez-Arellano, G., Terrazas, G. and Ratchev, S., 2019. Tool wear classification using time series imaging and deep learning. The International Journal of Advanced Manufacturing Technology, 104(9-12), pp.3647-3662.

[159] Cao, X.C., Chen, B.Q., Yao, B. and He, W.P., 2019. Combining translation-invariant wavelet frames and convolutional neural network for intelligent tool wear state identification. Computers in Industry, 106, pp.71-84.

[160] Li XQ, Wong YS, Nee AYC. Tool wear and chatter detection using the coherence function of two crossed accelerations. Int J Mach Tools Manuf 1997;37:425–35.

[161] Choi T, Shin YC. On-Line Chatter Detection Using Wavelet-Based Parameter Estimation. J Manuf Sci Eng 2003;125:21.

[162] Yang F, Zhang B, Yu J. Chatter suppression with multiple time-varying parameters in turning. J Mater Process Technol 2003;141:431–8.

[163] Nair U, Krishna BM, Namboothiri VNN, Nampoori VPN. Permutation entropy based real-time chatter detection using audio signal in turning process. Int J Adv Manuf Technol 2010;46:61–8.

[164] Kwak JS. Application of wavelet transform technique to detect tool failure in turning operations. Int J Adv Manuf Technol 2006;28:1078–83.

[165] Neslušan M, Mičieta B, Mičietová A, Čilliková M, Mrkvica I. Detection of tool breakage during hard turning through acoustic emission at low removal rates. Meas J Int Meas Confed 2015;70:1–13.

[166] Li, L. and Steen, W.M., 1990, August. In-process clad quality monitoring using optical method. In Laser-Assisted Processing II (Vol. 1279, pp. 89-101). International Society for Optics and Photonics.

[167] Voelkel, D.D. and Mazumder, J., 1990. Visualization of a laser melt pool. Applied Optics, 29(12), pp.1718-1720.

[168] Hu, D. and Kovacevic, R., 2003. Sensing, modeling and control for laser-based additive manufacturing. International Journal of Machine Tools and Manufacture, 43(1), pp.51-60.

[169] Berumen, S., Bechmann, F., Lindner, S., Kruth, J.P. and Craeghs, T., 2010. Quality control of laser-and powder bed-based Additive Manufacturing (AM) technologies. Physics procedia, 5, pp.617-622.

[170] Lott, P., Schleifenbaum, H., Meiners, W., Wissenbach, K., Hinke, C. and Bültmann, J., 2011. Design of an optical system for the in situ process monitoring of selective laser melting (SLM). Physics Procedia, 12, pp.683-690.

[171] Krauss, H., Eschey, C. and Zaeh, M., 2012, August. Thermography for monitoring the selective laser melting process. In Proceedings of the Solid Freeform Fabrication Symposium (pp. 999-1014).

[172] Craeghs, T., Clijsters, S., Kruth, J.P., Bechmann, F. and Ebert, M.C., 2012. Detection of process failures in layerwise laser melting with optical process monitoring. Physics Procedia, 39, pp.753-759.

[173] Montazeri, M. and Rao, P., 2018. Sensor-Based Build Condition Monitoring in Laser Powder Bed Fusion Additive Manufacturing Process Using a Spectral Graph Theoretic Approach. Journal of Manufacturing Science and Engineering, 140(9), p.091002.

[174] Barua, S., Liou, F., Newkirk, J. and Sparks, T., 2014. Vision-based defect detection in laser metal deposition process. Rapid Prototyping Journal, 20(1), pp.77-85.

[175] Clijsters, S., Craeghs, T., Buls, S., Kempen, K. and Kruth, J.P., 2014. In situ quality control of the selective laser melting process using a high-speed, real-time melt pool monitoring system. The International Journal of Advanced Manufacturing Technology, 75(5-8), pp.1089-1101.

[176] Mireles, J., Terrazas, C., Gaytan, S.M., Roberson, D.A. and Wicker, R.B., 2015. Closed-loop automatic feedback control in electron beam melting. The International Journal of Advanced Manufacturing Technology, 78(5-8), pp.1193-1199.

[177] Zenzinger, G., Bamberg, J., Ladewig, A., Hess, T., Henkel, B. and Satzger, W., 2015, March. Process monitoring of additive manufacturing by using optical tomography. In AIP Conference Proceedings (Vol. 1650, No. 1, pp. 164-170). AIP.

[178] Toeppel, T., Schumann, P., Ebert, M.C., Bokkes, T., Funke, K., Werner, M., Zeulner, F., Bechmann, F. and Herzog, F., 2016. 3D analysis in laser beam melting based on real-time process monitoring. In Mater Sci Technol Conf.

[179] List III, F.A., Dinwiddie, R.B., Carver, K. and Gockel, J.E., 2017. Melt-Pool Temperature and Size Measurement During Direct Laser Sintering (No. ORNL/TM-2017/4). Oak Ridge National Lab.(ORNL), Oak Ridge, TN (United States).

[180] Khanzadeh, M., Tian, W., Yadollahi, A., Doude, H.R., Tschopp, M.A. and Bian, L., 2018. Dual process monitoring of metal-based additive manufacturing using tensor

decomposition of thermal image streams. Additive Manufacturing, 23, pp.443-456.
[181] Khanzadeh, M., Chowdhury, S., Tschopp, M.A., Doude, H.R., Marufuzzaman, M. and Bian, L., 2018. In-situ monitoring of melt pool images for porosity prediction in directed energy deposition processes. IISE Transactions, pp.1-19.
[182] Zhang, Y., Hong, G.S., Ye, D., Zhu, K. and Fuh, J.Y., 2018. Extraction and evaluation of melt pool, plume and spatter information for powder-bed fusion AM process monitoring. Materials & Design, 156, pp.458-469.
[183] Gao, X., Wang, L., You, D., Chen, Z. and Gao, P.P., 2019. Synchronized monitoring of droplet transition and keyhole bottom in high power laser-MAG hybrid welding process. IEEE Sensors Journal.
[184] Xiong, J. and Zhang, G., 2013. Online measurement of bead geometry in GMAW-based additive manufacturing using passive vision. Measurement Science and Technology, 24(11), p.115103.
[185] Slotwinski, J.A., Garboczi, E.J. and Hebenstreit, K.M., 2014. Porosity measurements and analysis for metal additive manufacturing process control. Journal of research of the National Institute of Standards and Technology, 119, p.494.
[186] Rieder, H., Dillhöfer, A., Spies, M., Bamberg, J. and Hess, T., 2014, October. Online monitoring of additive manufacturing processes using ultrasound. In 11th European Conference on Non-Destructive Testing (ECNDT), Prague, Czech Republic, Oct (pp. 6-10).
[187] Zhang, B., Ziegert, J., Farahi, F. and Davies, A., 2016. In situ surface topography of laser powder bed fusion using fringe projection. Additive Manufacturing, 12, pp.100-107.
[188] DePond, P.J., Guss, G., Ly, S., Calta, N.P., Deane, D., Khairallah, S. and Matthews, M.J., 2018. In situ measurements of layer roughness during laser powder bed fusion additive manufacturing using low coherence scanning interferometry. Materials & Design, 154, pp.347-359.
[189] Ancona, A., Spagnolo, V., Lugara, P.M. and Ferrara, M., 2001. Optical sensor for real-time monitoring of CO 2 laser welding process. Applied Optics, 40(33), pp.6019-6025.

[190] Nicolosi, L., Tetzlaff, R., Abt, F., Blug, A., Carl, D. and Hofler, H., 2009, June. New CNN based algorithms for the full penetration hole extraction in laser welding processes: Experimental results. In 2009 International Joint Conference on Neural Networks (pp. 2256-2263). IEEE.
[191] Kim, C.H. and Ahn, D.C., 2012. Coaxial monitoring of keyhole during Yb: YAG laser welding. Optics & Laser Technology, 44(6), pp.1874-1880.
[192] You, D., Gao, X. and Katayama, S., 2014. Multisensor fusion system for monitoring high-power disk laser welding using support vector machine. IEEE Transactions on Industrial

Informatics, 10(2), pp.1285-1295.
[193] You, D., Gao, X. and Katayama, S., 2015. WPD-PCA-based laser welding process monitoring and defects diagnosis by using FNN and SVM. IEEE Transactions on Industrial Electronics, 62(1), pp.628-636.
[194] Luo, M. and Shin, Y.C., 2015. Vision-based weld pool boundary extraction and width measurement during keyhole fiber laser welding. Optics and Lasers in Engineering, 64, pp.59-70.
[195] Luo, M. and Shin, Y.C., 2015. Estimation of keyhole geometry and prediction of welding defects during laser welding based on a vision system and a radial basis function neural network. The International Journal of Advanced Manufacturing Technology, 81(1-4), pp.263-276.
[196] Xu, J., Rong, Y., Huang, Y., Wang, P. and Wang, C., 2018. Keyhole-induced porosity formation during laser welding. Journal of Materials Processing Technology, 252, pp.720-727.
[197] Tang, L. and Landers, R.G., 2010. Melt Pool Temperature Control for Laser Metal Deposition Processes—Part I: Online Temperature Control. Journal of manufacturing science and engineering, 132(1), p.011010.
[198] Song, L., Bagavath-Singh, V., Dutta, B. and Mazumder, J., 2012. Control of melt pool temperature and deposition height during direct metal deposition process. The International Journal of Advanced Manufacturing Technology, 58(1-4), pp.247-256.
[199] Wang, Q., Li, J., Gouge, M., Nassar, A.R., Michaleris, P.P. and Reutzel, E.W., 2017. Physics-based multivariable modeling and feedback linearization control of melt-pool geometry and temperature in directed energy deposition. Journal of Manufacturing Science and Engineering, 139(2), p.021013.
[200] Hofman, J.T., Pathiraj, B., Van Dijk, J., De Lange, D.F. and Meijer, J., 2012. A camera based feedback control strategy for the laser cladding process. Journal of Materials Processing Technology, 212(11), pp.2455-2462.
[201] Gockel, J., Beuth, J. and Taminger, K., 2014. Integrated control of solidification microstructure and melt pool dimensions in electron beam wire feed additive manufacturing of Ti-6Al-4V. Additive Manufacturing, 1, pp.119-126.
[202] Farshidianfar, M.H., Khajepour, A. and Gerlich, A., 2016. Real-time control of microstructure in laser additive manufacturing. The International Journal of Advanced Manufacturing Technology, 82(5-8), pp.1173-1186.
[203] Hornik, K., Stinchcombe, M. and White, H., 1989. Multilayer feedforward networks are universal approximators. Neural networks, 2(5), pp.359-366.

[204] Bonamente M. (2017) Functions of Random Variables and Error Propagation. In: Statistics and Analysis of Scientific Data. Graduate Texts in Physics. Springer, New York, NY

[205] Frederic, P. and Lad, F., 2008. Two moments of the logitnormal distribution. Communications in Statistics—Simulation and Computation®, 37(7), pp.1263-1269.

[206] Hershey, J.R. and Olsen, P.A., 2007, April. Approximating the Kullback Leibler divergence between Gaussian mixture models. In 2007 IEEE International Conference on Acoustics, Speech and Signal Processing-ICASSP'07 (Vol. 4, pp. IV-317). IEEE.

[207] Golub, G.H. and Van Loan, C.F., 2012. Matrix computations (Vol. 3). JHU press.

[208] Runnalls, A.R., 2007. Kullback-Leibler approach to Gaussian mixture reduction. IEEE Transactions on Aerospace and Electronic Systems, 43(3), pp.989-999.

[209] Muscolino, G., Ricciardi, G. and Vasta, M., 1997. Stationary and non-stationary probability density function for non-linear oscillators. International Journal of Non-Linear Mechanics, 32(6), pp.1051-1064.

[210] Berning, A.W., Girard, A., Kolmanovsky, I. and D'Souza, S.N., 2019. Rapid uncertainty propagation and chance-constrained path planning for small unmanned aerial vehicles. Advanced Control for Applications: Engineering and Industrial Systems, p.e23.

[211] Luukkonen, T., 2011. Modelling and control of quadcopter. Independent research project in applied mathematics, Espoo, 22.

[212] Tüfekci, P., 2014. Prediction of full load electrical power output of a base load operated combined cycle power plant using machine learning methods. International Journal of Electrical Power & Energy Systems, 60, pp.126-140.

[213] UC Irvine Machine Learning Repository, http://archive.ics.uci.edu/ml/datasets/Combined +Cycle+Power+Plant, accessed on Apr, 2020.

[214] LeCun, Y., 1998. The MNIST database of handwritten digits. http://yann.lecun. com/exdb/mnist/, Accessed on Aug, 2020.

[215] Simard, P.Y., Steinkraus, D. and Platt, J.C., 2003, August. Best practices for convolutional neural networks applied to visual document analysis. In Icdar (Vol. 3, No. 2003).

[216] Ahmed, N.A. and Gokhale, D.V., 1989. Entropy expressions and their estimators for multivariate distributions. IEEE Transactions on Information Theory, 35(3), pp.688-692.

[217] Qiao, Y. and Minematsu, N., 2010. A study on invariance of *f*-divergence and its application to speech recognition. IEEE Transactions on Signal Processing, 58(7), pp.3884-3890.

[218] Hershey, J.R. and Olsen, P.A., 2007, April. Approximating the Kullback Leibler divergence between Gaussian mixture models. In 2007 IEEE International Conference on Acoustics, Speech and Signal Processing-ICASSP'07 (Vol. 4, pp. IV-317). IEEE.

[219] Rudin, W., 1976. Principles of mathematical analysis, 3rd ed. New York: McGraw-hill.

[220] Barber, D., 2012. Bayesian reasoning and machine learning. Cambridge University Press.

[221] Reutenauer, C. and Schützenberger, M.P., 1987. A formula for the determinant of a sum of matrices. letters in mathematical physics, 13(4), pp.299-302.

[222] Le Gland, F. and Mevel, L., 2000. Exponential forgetting and geometric ergodicity in hidden Markov models. Mathematics of Control, Signals and Systems, 13(1), pp.63-93.

[223] Psiaki, M.L., 2013. The blind tricyclist problem and a comparative study of nonlinear filters: A challenging benchmark for evaluating nonlinear estimation methods. IEEE Control Systems Magazine, 33(3), pp.40-54.

[224] Bottou, L., 2010. Large-scale machine learning with stochastic gradient descent. In Proceedings of COMPSTAT'2010 (pp. 177-186). Physica-Verlag HD.

[225] Anderson, M., Patwa, R. and Shin, Y.C., 2006. Laser-assisted machining of Inconel 718 with an economic analysis. International Journal of Machine Tools and Manufacture, 46(14), pp.1879-1891.

[226] Dandekar, C.R., Shin, Y.C. and Barnes, J., 2010. Machinability improvement of titanium alloy (Ti–6Al–4V) via LAM and hybrid machining. International Journal of Machine Tools and Manufacture, 50(2), pp.174-182.

[227] Liu, S. and Shin, Y.C., 2019. Additive manufacturing of Ti6Al4V alloy: A review. Materials & Design, 164, p.107552.

[228] Pellone, L., Inamke, G., Hong, K.M. and Shin, Y.C., 2019. Effects of Interface Gap and Shielding Gas on the Quality of Alloy AA6061 Fiber Laser Lap Weldings. Journal of Materials Processing Technology.

[229] Bay, H., Ess, A., Tuytelaars, T. and Van Gool, L., 2008. Speeded-up robust features (SURF). Computer vision and image understanding, 110(3), pp.346-359.

[230] Ioffe, S. and Szegedy, C., 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167.

[231] Kingma, D.P. and Ba, J., 2014. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

[232] Wang, Z., Liu, P., Ji, Y., Mahadevan, S., Horstemeyer, M.F., Hu, Z., Chen, L. and Chen, L.Q., 2019. Uncertainty quantification in metallic additive manufacturing through physics-informed data-driven modeling. Jom, 71(8), pp.2625-2634

PUBLICATIONS

[1] **Zhang, B.**, Katinas, C. and Shin, Y.C., 2018. "Robust tool wear monitoring using systematic feature selection in turning processes with consideration of uncertainties". *Journal of Manufacturing Science and Engineering*, 140(8). https://doi.org/10.1115/1.4040267

[2] **Zhang, B.** and Shin, Y.C., 2018. "A multimodal intelligent monitoring system for turning processes". *Journal of Manufacturing Processes*, 35, pp.547-558. https://doi.org/10.1016/j.jmapro.2018.08.021

[3] **Zhang, B.**, Liu, S. and Shin, Y.C., 2019. "In-Process monitoring of porosity during laser additive manufacturing process". *Additive Manufacturing*, 28, pp.497-505. https://doi.org/10.1016/j.addma.2019.05.030

[4] **Zhang, B.**, Hong, K.M. and Shin, Y.C., 2020. "Deep-learning-based porosity monitoring of laser welding process". *Manufacturing Letters*, 23, pp. 62-66 https://doi.org/10.1016/j.mfglet.2020.01.001

[5] **Zhang, B.**, Katinas, C. and Shin, Y.C., 2020. "Robust wheel wear monitoring system for cylindrical traverse grinding". *IEEE/ASME Transactions on Mechatronics*, 25(5), pp. 2220-2229. https://doi.org/10.1109/TMECH.2020.3007047

[6] **Zhang, B.** and Shin, Y.C., 2021. "A data-driven approach of Takagi-Sugeno fuzzy control of unknown nonlinear systems". *Applied Sciences*, 11(1), pp. 62-76. https://doi.org/10.3390/app11010062

[7] **Zhang, B**. and Shin, Y.C., 2021. "An Adaptive Gaussian Mixture Method for Nonlinear Uncertainty Propagation in Neural Networks". *Neurocomputing* , 485(7), pp. 170-183 https://doi.org/10.1016/j.neucom.2021.06.007

[8] **Zhang, B.** and Shin, Y.C., 2021. "An Adaptive Gaussian Mixture Filter for Nonlinear State Estimation". *IEEE Transactions on Cybernetics* (under review)

[9] **Zhang, B.** and Shin, Y.C., 2021. "Data-Driven Phase Recognition of Steels for Use in Mechanical Property Prediction". *Manufacturing Letters* (under review)

[10] **Zhang, B**. and Shin, Y.C., 2021. "A Probabilistic Neural Network for Uncertainty Prediction with Applications to Manufacturing Process Monitoring". Applied Soft Computing (under review)