A MultiStage Approach for Knowledge-Guided Predictions With Application to Additive Manufacturing

Seokhyun Chung, Cheng-Hao Chou, Xiaozhu Fang, Raed Al Kontar, and Chinedum Okwudire, *Member, IEEE*

***Abstract*— Inspired by sequential additive manufacturing operations, we consider prediction tasks arising in processes that comprise of sequential sub-operations and propose a multi-stage inference procedure that exploits prior knowledge of the operational sequence. Our approach decomposes a data-driven model into several easier problems each corresponding to a sub-operation and then introduces a Bayesian inference procedure to quantify and propagate uncertainty across operational stages. We also complement our model with an approach to incorporate physical knowledge of the output of a sub-operation which is often more practical in reality relative to understanding the physics of the entire process. Comprehensive simulations and two case studies on additive manufacturing show that the proposed framework provides well-quantified uncertainties and superior predictive accuracy compared to a single-stage predictive approach.**

***Note to Practitioners*—This paper is motivated by sequential operations that often occur in manufacturing processes. For example, several additive manufacturing processes consist of multiple sequential steps, e.g., printing, washing, and curing in stereolithography, or printing, debinding, and sintering in binder jetting. In such settings, a complex data-driven model that blindly throws all given data into a single predictive model might not be optimal. To this end, we propose a multi-stage inference procedure that decomposes the problem into easier sub-problem using the prior knowledge of the operational sequence, and propagates uncertainty across stages using Bayesian neural networks. Here we note that even if sequential operations are not existent in reality, one may conceptually decompose a complex system into simpler pieces and exploit our procedure. Also, our approach is able to incorporate physical knowledge of the output of a sub-operation.**

***Index Terms*— Multi-stage inference, additive manufacturing, Bayesian neural networks.**

Manuscript received February 13, 2022; accepted March 12, 2022. This article was recommended for publication by Associate Editor M.-H. Hung and Editor F.-T. Cheng upon evaluation of the reviewers' comments. This work was supported by CPS under Award 1931950. *(Corresponding author: Raed Al Kontar.)*

Seokhyun Chung and Raed Al Kontar are with the Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, MI 48109 USA (e-mail: alkontar@umich.edu).

Cheng-Hao Chou, Xiaozhu Fang, and Chinedum Okwudire are with the Department of Mechanical Engineering, University of Michigan, Ann Arbor, MI 48109 USA.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TASE.2022.3160420.

Digital Object Identifier 10.1109/TASE.2022.3160420

I. Introduction

FOR many complex systems nowadays, understanding the physical principles that guide the underlying process is an increasingly challenging task. As a result, data-driven approaches have risen as a powerful alternative (or complement) for physical guidance, to model and predict the evolution of a system and its operating state. Rapid advances in Internet of Things enabled systems has helped pave the way for such data-driven models, through facilitating the supply of large scale data; be it in size, dimension, data type, amongst others. Indeed, data-driven predictive analytics have seen many successes in various engineering fields. Some examples include fault detection of industrial motors [1], remaining useful life prediction of milling machines [2], to name a few.

Despite recent success, data-driven approaches are highly vulnerable to model mis-specifications and model inference challenges, specifically within highly complex processes. Data collected from such processes often exhibits high non-linearity, and faces the curse of dimensionality in the presence of many attributes and sub-processes [3]. Even with sufficient data, training data-driven model in such settings poses significant inference challenges: over-fitting, vulnerability to being trapped around bad local minima, slow convergence, and inscalability [4].

On the other hand, knowledge (or physics) based models exploit a priori scientific knowledge to deduce the relationship among physical variables in a system. Inspired by such models, a large body of work on knowledge-guided statistical models has been conducted to address the challenges arising in pure data-driven modeling [5]. Naturally, strategies incorporating physical knowledge with a data-driven model vary widely depending on systems' characteristics. In this study, our focus is on a system with sequential sub-operations. One common example is in additive manufacturing where it is not uncommon for a part to be produced via multiple sequential operations. For example, stereolithography [6] is an additive manufacturing process comprising three operations: printing, washing, and curing; binder jetting [7] is an increasingly popular additive manufacturing process that also involves three operations: printing, debinding, and sintering. Given knowledge of the operational sequence, a modeling strategy for such systems is to employ a data-driven model

for each sub-operation. Some examples can be found in different domains. [8] estimated a 3D human pose composed of top and bottom joints where each joint is modeled by a neural network (NN). [9] proposed compositional NNs that reflect prior knowledge on eye localization processes. Along this line, [10] proposed a knowledge-based NN for additive manufacturing where physical operations are linked based on a causal graph based on dimensional analysis. Despite promising results of knowledge-guided data-driven models for sequential operations, we see limitations of most existing studies in three aspects: (i) They do not consider predictive uncertainty that represents how reliable the prediction of a knowledge-guided model is. Indeed, this requires uncertainty propagation across stages which may be challenging. (ii) Knowledge is only accounted for through an architecture that resembles the sequential system. Yet, current literature cannot incorporate exact or inexact physical knowledge of a sub-operation, which is often more practical in reality relative to understanding the physics of the entire process. (iii) Literature only deals with cases where an operational sequence is explicitly recognized.

In this paper, we consider prediction tasks arising in processes that comprise of sequential sub-operations. Here a sequential structure does not have to be explicit. For instance, even if a system encompasses one process, one may conceptually decompose such operation to simpler pieces: often a set of base processes/components. We argue that a purely data-driven method that blindly throws all given data in a single predictive model might not be optimal. Rather, one may exploit prior knowledge of the operational sequence to decompose a data-driven model into several easier problems, each corresponding to a sub-operation. With this end goal in mind, we propose a knowledge-guided multi-stage inference framework that is able to propagate uncertainty and incorporate physical knowledge. We specifically focus on Bayesian neural networks (BNN) and propose a Bayesian inference procedure that quantifies uncertainties at each stage and propagates them to sequential stages. Our framework is also amenable to incorporating physical knowledge of the output of a sub-operation. Through extensive numerical studies on both simulated and real-world case studies on additive manufacturing we show that our method excels in improving predictive capabilities of sequential processes. Prior to highlighting our contributions, we start by real-world examples that motivate our approach.

A. Motivating Examples

The proposed multi-stage framework is mainly inspired by sequential operations that occur in a variety of manufacturing processes. For example, in the additive manufacturing process known as stereolithography (SLA) [6], a part undergoes three operations as shown in Fig. 1. The operations are: printing, where the physical shape of the object is produced by photopolymerization of a resin; washing in isopropyl alcohol, to remove excessive resin from the printed part; and curing by exposing the printed part to ultraviolet light to improve its mechanical properties. The final part dimension is affected by the part dimensions resulting from each operation. However,

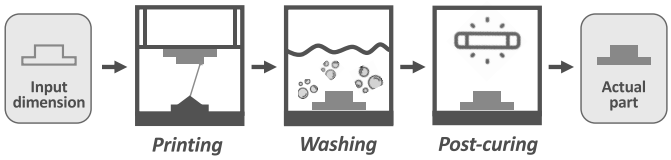

Fig. 1. Example of a stereolithography, an additive manufacturing process comprising a sequence of three operations.

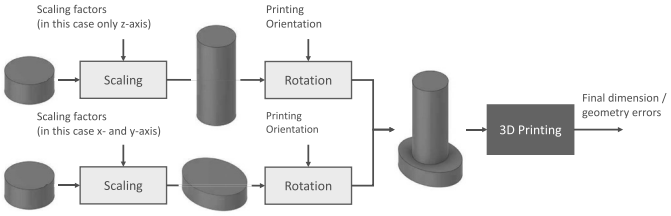

Fig. 2. Example of a case in additive manufacturing where a process is decomposed into multiple imaginary sequential operations.

for such complex systems with multiple affecting operations, a simple predictive model may fail to provide accurate predictions of the final part dimension due to its low representation power. On the other hand, using a more complex model is not a silver bullet either, because of the computational issues discussed earlier. However, if dimensional measurements are performed after each operation is finished, one may be able to exploit the knowledge of the sequential processes to model the effects of each operation using multiple simple models. The issues of using too simple or too complex models can thereby be prevented.

Moreover, the idea of imaginary operations, such as the geometry transformation of the desired parts, also motivates the usage of the proposed multi-stage framework. For example, as shown in Fig. 2, the bolt without thread can be decomposed into a long circular cylinder and a shorter elliptical cylinder; furthermore, these two cylinders can be regarded as a circular disk subject to different geometry transformations. Since it is expected that similar product shapes, e.g., disk, cylinders, and elliptical cylinders, may share similar characteristics due to similar dynamic response and thermal distribution etc., the multi-stage model can also be useful in this situation that the earlier stages model the simpler geometry and the later stages consider further geometry transformation effects. These imaginary processes also decompose a complex system into simpler pieces, reducing the model complexity.

B. Contributions

We summarize our contributions in this article as follows.

- We propose a predictive analytics framework for settings that involve sequential processes. Our approach is also able to incorporate physical knowledge of the output of a sub-operation which is often more practical in reality relative to understanding the physics of the entire process. Despite our focus on NN, our approach is amenable to different statistical models.
- We propose a principled way to propagate uncertainty between stages. Our approach is based on stochastic gradient Markov chain Monte-Carlo (SGMCMC) [11]

and is compatible with any Bayesian probabilistic model for regression. To the best of our knowledge, this is the first work exploring uncertainty propagation between sequential BNNs.
- We extensively evaluate the proposed model using both simulated and real-world data. The results show that the model outperforms benchmark models as well as is capable of providing well-quantified predictive uncertainty.

The remainder of the paper is structured as follows. In Sec. II, we review related work on knowledge-guided statistical models and BNNs. In Sec. III, we discuss the basic framework of the proposed knowledge-guided multi-stage model. In Sec. IV, we comprehensively examine the model using simulations. Two additive manufacturing case studies are then presented in Sec. V. Finally, Sec. VI concludes the paper with a brief discussion.

II. Related Work

A. Knowledge-Guided Statistical Models

While knowledge based models provide explanations on intrinsic physical principles, they often fail in practice as real systems possess inherent complexity that cannot be easily understood. On the other hand, data-driven models introduce a statistical model that infers the relationship using collected data. They can capture inherent complexity exhibited in the data, whereas hardly serve causality to intuitively understand intrinsic principles and are vulnerable to model misspecifications.

Taking the best of both worlds, a large body of work on knowledge-guided statistical models has been conducted. Related studies are categorized based on the strategy for embedding knowledge in data-driven models [5]. One strategy is to maneuver a statistical model towards physically consistent solutions. Such strategies include: (i) setting a knowledge-based solution as the starting point of the inference algorithm [12]. (ii) Placing a knowledge-guided prior under a Bayesian framework [13]–[15]. (iii) Regularizing the objective in a frequentist framework. In such approaches, the objective function of a data-driven model is penalized with a term that reflects domain knowledge. Such approaches have been explored in climate science [16], genome informatics [17], geology [18] and fragility analysis [19], amongst many others. Readers interested in the comprehensive review of knowledge-guided statistical models are referred to [5]. (iv) Amongst others, designing a model architecture in accordance with domain knowledge is the incorporating strategy most closely related to the multi-stage framework. In particular, there have been some studies where the architecture of a NN is modeled to reflect prior knowledge on the system. For instance, [20] investigated the prediction of lake temperature, proposing a recurrent NN with nodes representing the energy and mass conservation of a lake obtained by physical simulations. The studies, introduced earlier, on knowledge-guided NNs to model eye localization [9] or an additive manufacturing process [10] also fall into this category.

In the context of a knowledge-guided multi-stage framework where information propagates across stages, the stream of variation (SOV) analysis [21] for multi-station machining processes is perhaps the most closely related to our approach. The SOV analysis models the variations of products potentially propagated through multi-stage manufacturing processes, which state-space models can successfully describe [22]–[24]. However, our proposed approach is different from the SOV modeling in that: (i) it models each stage using a data-driven model (e.g., BNN); (ii) it can conceptually decompose one complex operation into simpler imaginary sub-operations; (iii) it propagates predictive uncertainty from BNN but not dimensional variations.

B. Bayesian NN

There have been recent efforts to rethink NN from a Bayesian perspective. This research was driven by the need to quantify uncertainty within NNs and the often lack of reliability obtained from point NN solutions obtained by vanilla stochastic gradient descent (SGD) [25]–[27]. Indeed, recent research has shown that NN can be extremely confident with their predictions yet extremely wrong [28]. To this end, BNNs [29], [30] have received significant attention. A BNN simply places a prior on the weight parameters and a posterior distribution is inferred given training data. Through this posterior, BNNs are capable of providing a predictive distribution rather than a single prediction value. Interestingly, many studies on BNNs have reported that the predictive mean often achieves better generalization than a NN in a variety of tasks. This mean is obtained through integrating out over the posterior. Hence instead of betting on one possible solution, BNNs integrate over multiple hypothesis weighted by their posterior probabilities. Such Bayesian model averaging have been linked to flatter solution; which is a possible explanation of the improved generalization [31].

However, the primary difficulty in BNNs is the intractability of the posterior distribution. Recent studies have proposed inference approaches that circumvent this challenge. One category makes use of variational inferences (VI) [32]. VI finds a surrogate of posterior distribution through minimizing Kullback-Leibler (KL) divergence between the original posterior and the proxy. Hence it reformulates the problem from an optimization perspective. In VI, the reparametrization trick [33], [34] is commonly used to derive stochastic gradients [35]. Another category includes posterior sampling methods. To avoid the heavy computation required in Markov Chain Monte-Carlo based sampling, SGMCMC has been proposed. For instance, the seminal work by [36] studied the connection between Langevin dynamics [37] and the search trajectory of a SGD when finding the maximum a posteriori (MAP): maximizing $P(\boldsymbol{\theta}|\mathcal{D}) \propto P(\boldsymbol{\theta})P(\mathcal{D}|\boldsymbol{\theta})$. Under mild conditions, they show that the parameters θ over the path of SGD updates, when injected with Gaussian noise following Langevin dynamics, can be used as the samples from the posterior $P(\boldsymbol{\theta}|\mathcal{D})$. Inheriting the scalability of stochastic gradient methods, they proposed a scalable sampling approach called stochastic gradient Langevin dynamics (SGLD). Readers interested in the comprehensive review on the inference of BNNs are referred to [38]. Here we should note that despite the

advances in BNN, their investigation and application within the engineering field remains limited.

III. Knowledge-Guided Multi-Stage Inferences via BNN

In this section, we discuss the proposed knowledge-guided multi-stage inference procedure. In Sec. III-A, we start by highlighting single-stage models. We then present the basic framework for our approach in Sec. III-B. Sec. III-C considers incorporating exact or inexact knowledge on a sub-operation. Sec. III-D introduces a Bayesian formulation of our framework for uncertainty quantification and propagation.

A. Data-Driven Models for Sequential Operations

Assume a dataset with N observations collected from a system composed of M inherent sequential sub-operations, denoted by $\mathcal{D} \equiv (\mathbf{X}, \mathbf{\Lambda}, \mathbf{Y})$, where $\mathbf{X} \in \mathbb{R}^{N\times d}$ represents the initial input of the system, $\mathbf{Y} \in \mathbb{R}^{N\times l}$ the final output, and $\mathbf{\Lambda} = (\mathbf{\Lambda}_1, \ldots, \mathbf{\Lambda}_M)$ where $\mathbf{\Lambda}_m \in \mathbb{R}^{N\times d_m}$ for $m = 1, \ldots, M$ are system input parameters specific to the m-th sub-operation. Note that we can collectively define $\mathbf{X}' = (\mathbf{X}, \mathbf{\Lambda}_1, \ldots, \mathbf{\Lambda}_M)$ as the input of the system, but we intentionally separate them for later development. A simple way to build a predictive model is

$$\mathbf{Y} = f(\mathbf{X}, \mathbf{\Lambda}_1, \ldots, \mathbf{\Lambda}_M; \boldsymbol{\theta}) + \epsilon, \tag{1}$$

where $f(\cdot; \boldsymbol{\theta})$ is a predictive model with parameters $\boldsymbol{\theta}$ and measurement noise ϵ. We refer to this model (1) as a single-stage model because the input-output relationship is established in one shot.

Now consider intermediate outputs from each stage are available and denote outputs at the m-th operation by $\mathbf{Y}_m \in \mathbb{R}^{N\times l_m}$. That is, while only $\mathbf{Y}$ is given in the model (1), we now have $\mathbf{Y}_1, \ldots, \mathbf{Y}_{M-1}$ as well as $\mathbf{Y} = \mathbf{Y}_M$ with $l = l_M$. An alternative single-stage model that uses all the given information is expressed as

$$\mathbf{Y}_M = f(\mathbf{X}, \mathbf{\Lambda}_1 \ldots \mathbf{\Lambda}_M, \mathbf{Y}_1, \ldots, \mathbf{Y}_{M-1}; \boldsymbol{\theta}) + \epsilon, \tag{2}$$

which we refer to as a single-stage model with additional information on intermediate outputs. The key idea behind (2), is to pool data from all stages into one large model. Inference for (1) and (2) then proceeds to finds $f(\cdot; \boldsymbol{\theta}^*)$ where $\boldsymbol{\theta}^*$ minimizes some empirical risk function. Those two models however do not exploit the sequential structure of the system. As aforementioned, a simple predictive model may not be capable of capturing the complex and high nonlinear relationship between inputs and outputs arising from the sequential operations. While a complex model is neither an absolute remedy, as it may overfit to training data, suffer from inscalability, get trapped in bad local minima, or even fail to converge.

B. The Basic Framework of Proposed Model

A simple idea to improve upon the models above is to exploit prior domain knowledge to decompose the problem into easier yet interrelated sub-problems. Mathematically, we inductively model the m-th operation as

$$\begin{aligned} \hat{\mathbf{Y}}_0 &= \mathbf{X}, \\ \mathbf{Y}_m &= f_m(\hat{\mathbf{Y}}_{m-1}, \mathbf{\Lambda}_m; \boldsymbol{\theta}_m) + \epsilon_m \ \text{ for } m \in \{1, \ldots, M\} \end{aligned} \tag{3}$$

where $\hat{\mathbf{Y}}_m = f_m(\hat{\mathbf{Y}}_{m-1}, \mathbf{\Lambda}_m; \boldsymbol{\theta}_m)$ for $m \in \{1, \ldots, M\}$ and ϵ_m is the measurement noise at the m-th operation. In this inductive model, the output from a previous model is fed to the next model. At each stage, we estimate $\boldsymbol{\theta}_m$ that minimizes an empirical risk defined through the loss function $L_m(\cdot, \cdot)$ at the m-th stage: $\boldsymbol{\theta}_m^* = \arg\min_{\boldsymbol{\theta}_m} L_m\big(f_m(\hat{\mathbf{Y}}_{m-1}, \mathbf{\Lambda}_m; \boldsymbol{\theta}_m), \mathbf{Y}_m\big)$.

Once model parameters $\boldsymbol{\theta}_m^*$ are estimated, predictions for a new observation $\mathbf{x}^*$ and system parameters $\boldsymbol{\lambda}_m^*$ are sequentially obtained by

$$\begin{aligned} \mathbf{y}_0^* &= \mathbf{x}^*, \\ \mathbf{y}_m^* &= f_m(\mathbf{y}_{m-1}^*, \boldsymbol{\lambda}_m^*; \boldsymbol{\theta}_m^*) \qquad \text{for } m \in \{1, \ldots, M\}, \end{aligned}$$

where $\mathbf{y}_M^*$ is the final prediction.

Fig. 3 shows a schematic illustration of the single-stage inference, single-stage inference with full information, and multi-stage inference. As shown in the figure, multi-stage inference decomposes one complicated process into several simpler sub-processes, thereby a model with less complexity can be employed for each sub-operation thus avoiding challenges induced in high-dimensional non-convex model learning. Also, such a model allows physical knowledge of sub-operations to be seamlessly integrated. This is highlighted in the following section.

C. Inverse Transformation: What If Physical Knowledge on the Input-Output Relationship of a Sub-Operation Is Known?

In practice, physical knowledge of sub-operations within the entire process is easier to extract compared to understanding that of the entire process (i.e. the transformation from $\mathbf{X}' \to \mathbf{Y}$). One key goal and motivation behind (3) is to address such situations. Through our multi-stage approach, one may apply inverse transformations to the final output to generate intermediate outputs when they are missing.

Without loss of generality, suppose that we do not have the intermediate output $\mathbf{Y}_m$ yet we have *partial* knowledge given as the physical formulas $\mathcal{P}_{m+1}(\cdot; \mathbf{\Lambda}_{m+1})$ that map stage specific input $(\hat{\mathbf{Y}}_m, \mathbf{\Lambda}_{m+1})$ to $\mathbf{Y}_{m+1}$. Then, we conduct inverse transformations, which are essentially the inverse function $\mathcal{P}_{m+1}^{-1}$, to generate

$$\tilde{\mathbf{Y}}_m = \mathcal{P}_{m+1}^{-1}(\mathbf{Y}_{m+1}; \mathbf{\Lambda}_{m+1}).$$

The generated output $\tilde{\mathbf{Y}}_m$ serves as a proxy of the unobserved output $\mathbf{Y}_m$. Here we say the knowledge is partial, because $\mathcal{P}_{m+1}^{-1}$ may not make $\tilde{\mathbf{Y}}_m$ perfectly independent of $\mathbf{\Lambda}_{m+1}$. Using the proxy $\tilde{\mathbf{Y}}_m$, we first train the m-th model given as

$$\tilde{\mathbf{Y}}_m = f_m(\hat{\mathbf{Y}}_{m-1}, \mathbf{\Lambda}_m; \boldsymbol{\theta}_m) + \epsilon_m$$

to estimate parameter $\boldsymbol{\theta}_m^*$ and thus we can get a pseudo output $\mathbf{Y}_m^{\text{pseudo}} = f_m(\hat{\mathbf{Y}}_{m-1}, \mathbf{\Lambda}_m; \boldsymbol{\theta}_m^*)$. Then, we train the $(m+1)$-th

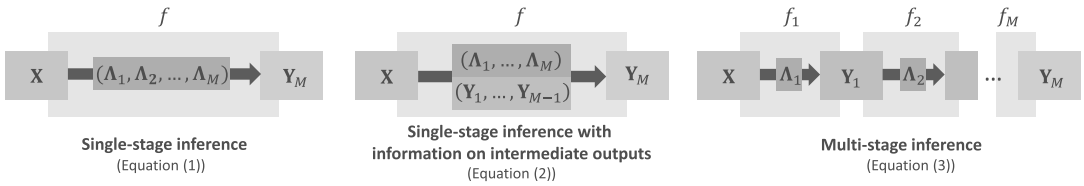

Fig. 3. An illustration of Multi-stage vs. single-stage inference with or without collecting outputs from intermediate operations.

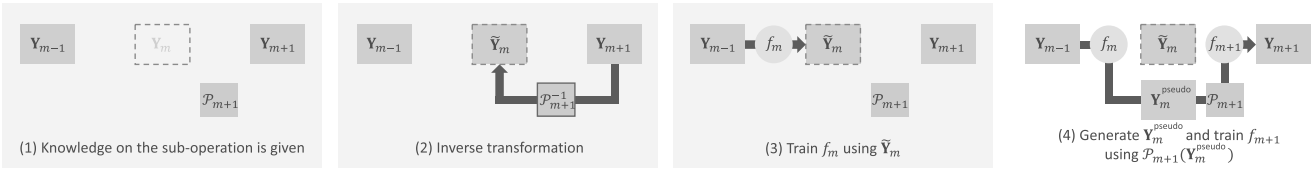

Fig. 4. The process of inverse transformation with a multi-stage framework.

model represented as

$$\mathbf{Y}_{m+1} = f_{m+1}\Big(\mathcal{P}_{m+1}\Big(\mathbf{Y}_m^{\text{pseudo}}; \mathbf{\Lambda}_{m+1}\Big), \mathbf{\Lambda}_{m+1}; \boldsymbol{\theta}_{m+1}\Big) + \epsilon_{m+1}. \tag{4}$$

Fig. 4 illustrates the process of inverse transformations. Here we emphasize two modeling insights at the $(m+1)$-th stage. First, we infer f_{m+1} instead of directly use $\mathcal{P}_{m+1}(\mathbf{Y}_m^{\text{pseudo}}, \mathbf{\Lambda}_{m+1})$. This is due to the potential incompleteness of the knowledge $\mathcal{P}_{m+1}(\cdot, \mathbf{\Lambda}_{m+1})$. The model $f_{m+1}(\cdot)$ can infer what $\mathcal{P}_{m+1}(\cdot, \mathbf{\Lambda}_{m+1})$ may miss out on the underlying truth. Second, we train the model f_{m+1} using $\mathcal{P}_{m+1}(\mathbf{Y}_m^{\text{pseudo}}, \mathbf{\Lambda}_{m+1})$ instead of directly using the pseudo output $\mathbf{Y}_m^{\text{pseudo}}$. By doing so, we can further exploit the knowledge from $\mathcal{P}_{m+1}$ in the inference of f_{m+1}.

D. Uncertainty Quantification Using BNN

Given our multi-stage framework, a key remaining challenge is quantifying uncertainty and propagating it through the stages. To this end, we propose a Bayesian extension for our multi-stage framework. For the m-th stage, denote the dataset by $\mathcal{D}_m \equiv (\hat{\mathbf{Y}}_{m-1}, \mathbf{\Lambda}_m, \mathbf{Y}_m)$. At each stage instead of finding $\boldsymbol{\theta}_m^*$, a Bayesian treatment first places a prior distribution $P(\boldsymbol{\theta}_m)$ and then infers a posterior distribution $P(\boldsymbol{\theta}_m|\mathcal{D}_m)$. This Bayesian inference is conducted at all stages for $m = 1, \ldots, M$.

Now assume that $P(\boldsymbol{\theta}_m|\mathcal{D}_m)$ are known for all $m \in \{1, \ldots, M\}$. Inference for this posterior will be discussed shortly. After inference, uncertainty propagation and quantification at a new input point $(\mathbf{x}^*, \boldsymbol{\lambda}^*)$ is done sequentially as shown in Alg. 1. Recall, $\boldsymbol{\lambda}^* = (\boldsymbol{\lambda}_1^*, \cdots, \boldsymbol{\lambda}_M^*)$.

The underlying idea behind Alg. 1 is to propagate a predictive distribution across stages. At the first stage, we take T samples from the posterior distribution $\boldsymbol{\theta}_1^{(t)} \sim P(\boldsymbol{\theta}_1|\mathcal{D}_1)$ for $t = 1, \ldots, T$. We then obtain the Monte-Carlo samples for predictions $\mathbf{y}_{1,t}^* = f_1(\mathbf{x}^*, \boldsymbol{\lambda}_1^*; \boldsymbol{\theta}_1^{(t)})$ for $t = 1, \ldots, T$. Using the samples, the predictive distribution can be approximated to $\mathcal{F}_m$. One example of approximating distributions is a Gaussian distribution

$$\mathcal{F}_1 := \mathcal{N}\big(\hat{\boldsymbol{\mu}}_1, \hat{\mathbf{\Sigma}}_1\big), \tag{5}$$

where $\hat{\boldsymbol{\mu}}_1 = \frac{1}{T}\sum_{t=1}^T \mathbf{y}_{1,t}^*$ and $\hat{\mathbf{\Sigma}}_1 = \frac{1}{T}\sum_{t=1}^T (\mathbf{y}_{1,t}^* - \hat{\boldsymbol{\mu}}_1)(\mathbf{y}_{1,t}^* - \hat{\boldsymbol{\mu}}_1)^\top$.

Algorithm 1 Uncertainty Propagation

Input: Posteriors $P(\boldsymbol{\theta}_1|\mathcal{D}_1), \ldots, P(\boldsymbol{\theta}_M|\mathcal{D}_M)$ and test inputs $(\mathbf{x}^*, \boldsymbol{\lambda}^*)$

Output: Predictive distribution $\mathcal{F}_M$

```
# The first stage
```

Sample $\boldsymbol{\theta}_1^{(t)} \sim P(\boldsymbol{\theta}_1|\mathcal{D}_1)$ for $t = 1, \ldots, T$.
Get predictions $\mathbf{y}_{1,t}^* = f_1(\mathbf{x}^*, \boldsymbol{\lambda}_1^*; \boldsymbol{\theta}_1^{(t)})$ for $1, \ldots, T$.
Approximate the distribution of $\mathbf{y}_{1,t}^*$ to $\mathcal{F}_1$.

```
# The second and later stages
```

for $m = 2, \ldots, M$ **do**
  Sample $\mathbf{z}_{m-1,s}^* \sim \mathcal{F}_{m-1}$ for $s = 1, \ldots, S$.
  Sample $\boldsymbol{\theta}_m^{(t)} \sim P(\boldsymbol{\theta}_m|\mathcal{D}_m)$ for $1, \ldots, T$.
  Get predictions $\mathbf{y}_{m,t,s}^* = f_m(\mathbf{z}_{m-1,s}^*, \boldsymbol{\lambda}_m^*; \boldsymbol{\theta}_m^{(t)})$ for $s = 1, \ldots, S$ and $1, \ldots, T$.
  Approximate the distribution of $\mathbf{y}_{m,t,s}^*$ to $\mathcal{F}_m$.
end for
return $\mathcal{F}_M$

We now feed the outputs from the first stage to the second stage. Here, the key idea is that, instead of $\hat{\boldsymbol{\mu}}_1$, we draw S samples from $\mathcal{F}_1$ and deliver them to the next stage. Using the samples denoted as $\mathbf{z}_{1,s}^*$ for $s = 1, \ldots, S$, we can obtain S predictions for each parameter sample $\boldsymbol{\theta}_2^{(t)} \sim P(\boldsymbol{\theta}_2|\mathcal{D}_2)$ using $\mathbf{y}_{2,t,s}^* = f_2(\mathbf{z}_{1,s}^*, \boldsymbol{\lambda}_2^*; \boldsymbol{\theta}_2^{(t)})$ for $t = 1, \ldots, T$ and $s = 1, \ldots, S$. We then approximate $\mathcal{F}_2$ using $S \times T$ prediction samples from the second BNN. We again form a set of samples drawn from the approximate predictive distribution at the second stage to feed the next stage. For the m-th stage with $m \geq 2$, we inductively perform this procedure until the final predictive distribution is obtained.

In our approach we generate samples from an approximated distribution (e.g., Eq. (5)) and deliver them to the next stage. This allows models to deliver the information about predictive variance to the next stage. Here we should note that one may also feed the sampled outputs (e.g. $\mathbf{y}_{2,t,s}^*$) directly to the next stage. However, the number of predictive samples at the m-th stage will be $S \times T^m$. On the contrary, by (5), $S \times T$ predictive samples at the m-th stage as we only transfer a fixed number of samples at each stage. Here one may not need the approximation if m is small enough thereby evaluating $S \times T^m$

samples using a predictive model does not require significant computation. We finally note that approximations other than the Gaussian (5) can be used, such as a mixture of Gaussians or a non-parametric approximation.

E. Posterior Sampling in BNN

The main challenge remaining is estimating $P(\boldsymbol{\theta}_m|\mathcal{D}_m)$. In a BNN it is usually intractable to find $P(\boldsymbol{\theta}_m|\mathcal{D}_m)$, yet one may hope to extract some samples from it. One obvious sample is the MAP, which is obtained by solving:

$$\underset{\boldsymbol{\theta}_m}{\text{Minimize}} \quad -(\log P(\boldsymbol{\theta}_m) + \log P(\mathcal{D}_m|\boldsymbol{\theta}_m)). \tag{6}$$

In a BNN, $\log P(\mathcal{D}_m|\boldsymbol{\theta}_m)$ is usually given as the log Gaussian likelihood $-\frac{N}{2}\log\sigma^2 - \frac{1}{2\sigma^2}\sum_{i=1}^{N}([f_m(\hat{\mathbf{y}}_{m-1})]_i - [\mathbf{y}_m]_i)^2$ for a regression problem or the cross-entropy for a multi-class classification.

Solving for the MAP is usually done via SGD. At each iteration t, SGD takes a mini-batch $\{\mathcal{D}_m^{(t,i)}\}_{i=1,\dots,n}$ of size n and updates $\boldsymbol{\theta}_m^{(t)}$ as $\boldsymbol{\theta}_m^{(t+1)} = \boldsymbol{\theta}_m^{(t)} - \eta_t \Delta\boldsymbol{\theta}_m^{(t)}$:

$$\Delta\boldsymbol{\theta}_m^{(t)} = \frac{\eta_t}{2}\left(\nabla \log P(\boldsymbol{\theta}_m^{(t)}) + \frac{N}{n}\sum_{i=1}^{n}\nabla \log P(\mathcal{D}_m^{(t,i)}|\boldsymbol{\theta}_m^{(t)})\right) \tag{7}$$

where η_t denotes the step size at iteration t. It is known that for SGD, and under the conditions of decreasing step sizes with $\sum_{t=1}^{\infty}\eta_t = \infty$ and $\sum_{t=1}^{\infty}\eta_t^2 < \infty$, parameters obtained by iterating (7) converge to a critical point.

The issue here is that the MAP estimation only provides the mode of a posterior, while we need multiple samples from $P(\boldsymbol{\theta}_m|\mathcal{D}_m)$ to propagate uncertainty. Here SGLD [36] provides an elegant solution.

The intuition behind SGLD is that, by injecting a Gaussian noise to (7) that depends on the step size, $\epsilon_t \sim \mathcal{N}(0, \eta_t\mathbf{I})$, the parameter updates do not collapse to the critical points and they converge to samples from the true posterior distribution. As a result, one can use the parameters collected from the SGLD updates as posterior samples. Specifically, the update is written as

$$\begin{aligned}\Delta\boldsymbol{\theta}_m^{(t)} &= \frac{\eta_t}{2}\left(\nabla \log P(\boldsymbol{\theta}_m^{(t)}) + \frac{N}{n}\sum_{i=1}^{n}\nabla \log P(\mathcal{D}_m^{(t,i)}|\boldsymbol{\theta}_m^{(t)})\right) \\ &\quad + \tau\epsilon_t, \\ \epsilon_t &\sim \mathcal{N}(0, \eta_t\mathbf{I}),\end{aligned} \tag{8}$$

where τ is the hyperparameter representing a thermal noise [38].

Through this, and at the stage m, we are now able to gather T samples $\boldsymbol{\theta}_m^{(t)}$ for $t \in \{1, \dots, T\}$ from $P(\boldsymbol{\theta}_m|\mathcal{D}_m)$. Those samples address the last missing part of Alg. 1. In practice, we set burn-in iterations to T' with the total iterations $T' + T$ [36], thus we collect the T parameter samples $\{\boldsymbol{\theta}_m^{(T'+1)}, \dots, \boldsymbol{\theta}_m^{(T'+T)}\}$ from (8). Those samples can be regarded as the samples drawn from the posterior distribution $P(\boldsymbol{\theta}_m|\mathcal{D}_m)$, used in the step 1 and 6 in Alg. 1.

We should note that SGLD uses the same step size for all parameters. This often leads to slow convergence when the loss surface of BNN is highly variable in the different directions in parameter space. This phenomenon is also seen in NN and was an underlying reason behind the Adam optimizer [39] and more recent alternatives. To this end, one may also utilize a preconditioned SGLD [40] that uses adaptive step sizes instead. We refer the reader to the Appendix for additional details.

We finally remark that, within the SGLD framework, training samples $\hat{\mathbf{Y}}_m$ can be obtained by the posterior mean approximated through SGLD iterations. More specifically, we calculate

$$\hat{\mathbf{Y}}_m = \begin{cases} \dfrac{1}{T}\displaystyle\sum_{t=T'+1}^{T'+T} f_m(\mathbf{X}, \boldsymbol{\Lambda}_m; \boldsymbol{\theta}_m^{(t)}) & \text{if } m = 1 \\ \dfrac{1}{ST}\displaystyle\sum_{t=T'+1}^{T'+T}\sum_{s=1}^{S} f_m(\mathbf{Z}_{m-1,s}, \boldsymbol{\Lambda}_m; \boldsymbol{\theta}_m^{(t)}) & \text{o.w.,} \end{cases}$$

where $\mathbf{Z}_{m-1,s} \in \mathbb{R}^{N\times l_{m-1}}$ denotes the samples obtained similarly to Step 5 in Alg. 1 but evaluated at training inputs. Note carefully that $\hat{\mathbf{Y}}_m$ is calculated in the training phase, not in the testing phase (i.e., Alg. 1).

IV. Proof of Concept

We start with a proof of concept for our proposed model. First, we examine the multi-stage BNN on synthetic data. Next, we discuss the case where some physical knowledge is known. For all simulation studies, we compare three models. (i) Multi-(B)NN: the multi-stage (Bayesian) NN, which is our proposed multi-stage model, (ii) Single-(B)NN: the single-stage (Bayesian) NN (iii) Single-(B)NN-Add: the single-stage (Bayesian) NN with additional information on outputs from intermediate stages. Note that we use "(B)NN" to collectively refer to Bayesian NN and non-Bayesian NN. The goal of this comparison is to shed light on the benefits of a knowledge-guided multi-stage structure (e.g., through Multi-(B)NN vs. Single-(B)NN) as well as uncertainty estimation and generalization via BNN (e.g., by Multi-BNN vs Multi-NN). Note that we set the number of hidden units in single-stage models to be equivalent to the sum of the number of hidden units of all models in the multi-stage model for a fair condition in model flexibility.

A. Simulation Study 1: Multi-Stage Inference

1) Setup:

a) Data Generation: We assume that the generating process of the synthetic data is composed of five sequential operations. To generate samples, we construct the sequential data generating model given by $y_m = y_m^{\mathsf{true}} + \epsilon_m = f_m(y_{m-1}^{\mathsf{true}}, \lambda_m) + \epsilon_m$ for $m = 1, \dots, 5$ and $y_0^{\mathsf{true}} = x$, where $f_m(\cdot, \lambda_m) = \mathcal{GP}_m(\cdot)$ is a realization of a Gaussian process with a squared exponential kernel $k_{\mathsf{exp}}(x, x') = \gamma\,\mathsf{exp}\left(-\frac{1}{2l}(x - x')^2\right)$ with $l = 0.3$ and $\gamma = 1$ and y_m^{true} is the underlying truth at the stage m without the measurement error $\epsilon_m \sim \mathcal{N}(0, 0.1^2)$. We generate 30 input data points $x_i \sim \mathsf{Unif}(-1, 1)$, where $\mathsf{Unif}(\cdot,\cdot)$ denotes a uniform distribution. For simplicity, we assume the operation-specific inputs λ_m are independent of y_{m-1}^{true} for all m.

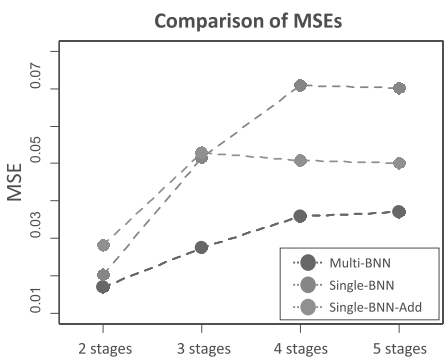

Fig. 5. The trend of average MSEs in the number of stages.

TABLE I
SUMMARY OF THE RESULTS ON SIMULATION STUDY 1

2 Stages	M-NN	M-BNN	S-NN-A	S-BNN-A	S-NN	S-BNN
Avg.	0.0176	**0.0171**	0.0272	0.0281	0.0200	0.0202
Std.	0.0061	0.0054	0.0114	0.0124	0.0115	0.0112
Time (sec.)	9.6	13.1	13.3	18.2	13.4	18.3
3 Stages	M-NN	M-BNN	S-NN-A	S-BNN-A	S-NN	S-BNN
Avg.	0.0280	**0.0276**	0.0522	0.0528	0.0539	0.0516
Std.	0.0261	0.0257	0.0387	0.0390	0.0784	0.0751
Time (sec.)	19.3	26.2	13.5	18.3	13.5	18.3
4 Stages	M-NN	M-BNN	S-NN-A	S-BNN-A	S-NN	S-BNN
Avg.	0.0365	**0.0360**	0.0508	0.0509	0.0720	0.0710
Std.	0.0432	0.0429	0.0446	0.0423	0.0945	0.0925
Time (sec.)	28.8	39.3	13.5	18.4	13.4	18.2
5 Stages	M-NN	M-BNN	S-NN-A	S-BNN-A	S-NN	S-BNN
Avg.	0.0373	**0.0370**	0.0496	0.0502	0.0715	0.0701
Std.	0.0281	0.0287	0.0384	0.0365	0.1113	0.1069
Time (sec.)	38.3	53.8	13.4	16.7	13.3	18.1

*M-(B)NN: Multi-(B)NN; S-(B)NN-A: Single-(B)NN-Add; S-(B)NN: Single-(B)NN.
**Best results are boldfaced.

b) Model Evaluation: We investigate the (i) predictive power using mean squared error (MSE) and (ii) uncertainty quantification capability of all benchmarked models. We collect 300 test inputs from $\mathsf{Unif}(-1, 1)$ for x to generate test outputs using the same data generating models above. The test outputs are collected from each stage to evaluate the m-stage Multi-(B)NN compared with Single-(B)NN (i.e., $y_m = f(x) + \epsilon$) and Single-(B)NN-add (i.e., $y_m = f(x, y_1, \ldots, y_{m-1}) + \epsilon$) for $m = 2, \ldots, 5$. Note that we use a predictive mean for BNNs to calculate their MSEs. Finally, we repeat the evaluation 20 times. Detailed settings for the model implementation and hyperparameters are deferred to the Appendix.

2) Simulation Results: The results are given in Fig. 5 and Table I. From the results, we can obtain several insights. First, the Multi-(B)NN outperforms Single-(B)NN and Single-(B)NN-Add in prediction accuracy. This confirms our intuition that knowledge-based structural modeling significantly helps increasing the model's prediction ability through decomposing it into multiple simpler problems. Second, we observe in Table I that the variances in the prediction of multi-stage models are less than the single-stage models, showing that predictions provided by our model are more robust. Here robustness is a direct consequence of the decomposition to simpler models and implies a reduced vulnerability to model mis-specifications, initialization and other optimization issues. In particular, the difference in prediction by single-stage models and Multi-(B)NN becomes more significant with a larger number of stages. This is intuitively understandable as more operations imply higher complexity in the input/output relationship and hence a greater challenge for single-stage inference. Finally, we see that Bayesian inference not only can quantify uncertainty but also provide better generalization. As shown in Table I, using a BNN can help better generalize to new data.

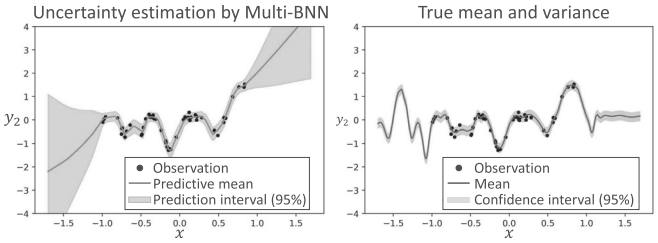

Fig. 6. Uncertainty quantification by Multi-BNN.

Regarding uncertainty quantification, we plot the predictive mean and variance for the final output y_2 of the two-stage model to check whether uncertainty is reasonably deduced. Specifically, we plot results for $x \in (-1.7, 1.7)$ in Fig. 6. We intentionally generate test data outside the training region $x \in (-1, 1)$ to see variance performance in unobserved regions. Comparing to the true confidence interval, Multi-BNN is able to reasonably quantify uncertainty. Further, the model clearly reflects the lack of information, via high variance, in extrapolated regions.

B. Simulation Study 2: Multi-Stage Inference With Inverse Transformation

We now study multi-stage inference with inverse transformations. The motivation of this simulation is from additive manufacturing. Consider 3D printing that stacks thin elliptical disks to produce a 3D object. The goal is to predict the shape of a manufactured disk, and its deviations from the target ellipse due to unknown physical factors like humidity and velocity of extruders. An ellipse is defined by scaling parameters (a, b) along major and minor axes and a rotation parameter ϕ (see Fig. 7). Given such parameters, we aim to predict the manufactured shape $y = f(x; a, b, \phi)$ where $x \in [0, 2\pi)$ using pairs of (a_i, b_i, ϕ_i) and y_i from shapes previously produced. Here, we hypothesize that there exists a sequence of inherent sub-operations in $f(\cdot)$: the ellipse shape is first affected by scaling and then rotation. Thus, in the first stage, we infer the model for scaling $f_1(\cdot; a, b)$ using unrotated shapes and their scaling parameters, and in the second stage, we infer the model for rotation $f_2(\cdot; \phi)$ that relates the output from the first model and rotation parameters to the corresponding rotated shapes.

Yet, since manufacturing an ellipse in reality is often done in one shot, we may not have the intermediate output: the dataset of actual shapes corresponding to the unrotated ellipses. However, we know $\mathcal{P}_2(\cdot, \phi)$. That is, we know the function that transforms the output from an unrotated ellipse to a rotated one. The inverse transformation of this function is simply given as

$$\mathcal{P}_2^{-1}(f(x); \phi) = f(x + \phi). \tag{9}$$

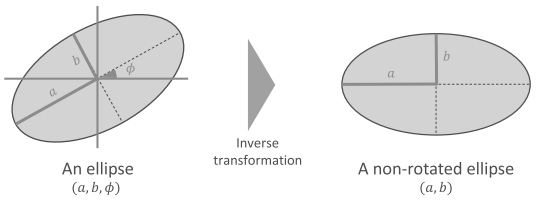

Fig. 7. Inverse transformation on an ellipse.

Thus from the final output, we can use (9) to generate the proxy of intermediate outputs and then employ a multi-stage inference framework.

1) Setup:

a) Data Generation: We generate the curve of $N = 300$ elliptical objects, evaluated at equally spaced 120 points $\mathbf{X} = [\mathbf{x}_i]_{i=1,\dots,N}$ where $\mathbf{x}_i = \left(0, \frac{3}{360}2\pi, \dots, \frac{357}{360}2\pi\right)^\top \in \mathbb{R}^{120\times 1}$ on the angular axis of the polar coordinate system. The system parameters are defined as $\mathbf{\Lambda} = (\mathbf{\Lambda}_1, \mathbf{\Lambda}_2)$ composed of the scale parameters $\mathbf{\Lambda}_1 = [\mathbf{a}, \mathbf{b}]$ with $\mathbf{a} = [a_i]^\top_{i=1,\dots,N}$ and $\mathbf{b} = [b_i]^\top_{i=1,\dots,N}$, and rotation parameter $\mathbf{\Lambda}_2 = \boldsymbol{\phi} = [\phi_i]^\top_{i=1,\dots,N}$. With these parameters, we generate the i-th output for the second operation $\mathbf{Y} = \mathbf{Y}_2 = [\mathbf{y}_{2,i}]_{i=1,\dots,N}$ with $\mathbf{y}_{2,i} \in \mathbb{R}^{120\times 1}$ for Case A and B as follows.

$$\text{Case A: } \mathbf{y}_{2,i} = \frac{a_i b_i}{\sqrt{a_i^2 \sin^2(\mathbf{x}_i - \phi_i) + b_i^2 \cos^2(\mathbf{x}_i - \phi_i)}} + a_i^{\frac{1}{3}} \sin(\mathbf{x}_i - \phi_i) + \epsilon, \tag{10}$$

$$\text{Case B: } \mathbf{y}_{2,i} = \frac{a_i b_i}{\sqrt{a_i^2 \sin^2(\mathbf{x}_i - \phi_i) + b_i^2 \cos^2(\mathbf{x}_i - \phi_i)}} + a_i^{\frac{1}{3}} \sin(\mathbf{x}_i - \frac{1}{2}\phi_i) + \epsilon. \tag{11}$$

We set the scaling parameters as $a_i \overset{i.i.d.}{\sim} \mathsf{Unif}(3,5)$, $b_i \overset{i.i.d.}{\sim} \mathsf{Unif}(1,3)$, the rotation parameter $\phi_i \overset{i.i.d.}{\sim} \mathsf{Unif}(0, \frac{1}{2}\pi)$, and Gaussian noise $\epsilon \overset{i.i.d.}{\sim} \mathcal{N}(0, 0.1^2)$. Note that the second term differentiates the cases. In Case A, the inverse transformation renders $\mathbf{Y}$ completely independent of the rotation parameter $\boldsymbol{\phi}$. This is ideal because the physical model $\mathcal{P}_2$ can perfectly represent its sub-operation. On the other hand, in Case B, the second term is still dependent on $\boldsymbol{\phi}$ after the inverse transformation is executed, implying that our physical knowledge is partial and incomplete. Case B will highlight the ability of the inverse transformation approach within our multi-stage framework to incorporate physical knowledge and compensate for its incompleteness.

b) Model Evaluation: First, we compare the models in terms of predictive accuracy. We generate the parameters a_i, b_i, ϕ_i for the test dataset by drawing 30 *i.i.d.* samples from the same distribution used for the training set. We evaluate the models using the average of the MSEs for predicted curves as an evaluation metric. This is expressed as $\frac{1}{120N}\sum_{i=1}^{N}\sum_{j=1}^{120}\left([\mathbf{Y}_i^*]_j - [\mathbf{Y}_i^{\mathsf{true}}]_j\right)^2$ where $\mathbf{Y}_i^*$ and $\mathbf{Y}_i^{\mathsf{true}}$ present the predicted vector for the i-th observation and the corresponding true values without noise, respectively. Next, we investigate how the estimated uncertainty obtained by our model evolves as moving further away from training observations, for both scaling and rotation parameters. We repeat the simulation 20 times. We finally note that Single-(B)NN-Add is excluded in this simulation as the intermediate output is not available. Detailed settings for model implementation and hyperparameters are deferred to the Appendix.

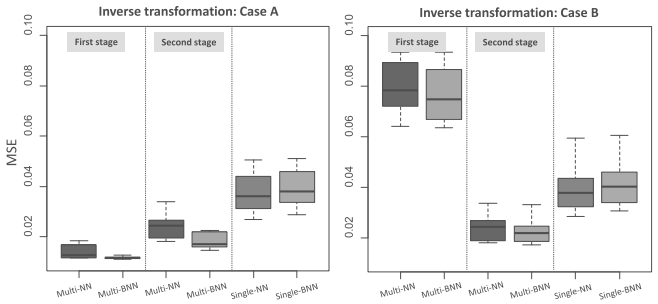

Fig. 8. Boxplots for MSEs of the comparative models with inverse transformation.

2) Simulation Results: Results are highlighted in Fig. 8. We can directly see that multi-stage (B)NNs outperform their single-stage counterparts consistently in both Case A and Case B. This sheds light on the advantage of exploiting physical knowledge through inverse transformations and also the benefit of a multi-stage framework even when a process is not sequential but can be conceptually divided into simpler sequential pieces. To further highlight this, boxplots in Case B show that the first stage does not predict well for the inferred intermediate outputs because of the incomplete knowledge on the inverse transformation. Despite that, multi-stage models eventually excel at the final prediction and outperform a single-stage approach. This is intuitively understandable, as the multi-stage framework in (4) can capture inherent association across the stages through the additional knowledge incorporated. This demonstrates that even if intermediate output prediction ($\mathbf{Y}_1^*$) is not very accurate, if they are more associated with the final output ($\mathbf{Y}_2$), they can improve predictive power.

We also provide plots in Fig. 9 for both interpolated results within the design space (column 1) and extrapolated results (columns 2-4). Here we either fix the rotation parameter $\phi = 0.25\pi$ (the first row) or the scaling parameter $(a, b) = (4, 2)$ (the second row). The results show that we can accurately predict within the interpolated region. Intriguingly, quite reasonable extrapolations are obtained in the scaling space whereas the extrapolation is quickly spoiled in the rotation space. We conjecture that the underlying reason is that the functional behaviour in extrapolated regions of the rotation space is largely different compared to that within the design space.

V. Case Studies: Additive Manufacturing

We apply the proposed framework to predict the dimensional errors of additively manufactured parts via two case studies. In the first case study, the parts are decomposed into a sequence of imaginary operations which are modeled using the proposed framework. In the second case study, the parts are produced using an additive manufacturing process that physically involves three sequential processes which are modeled using the proposed framework.

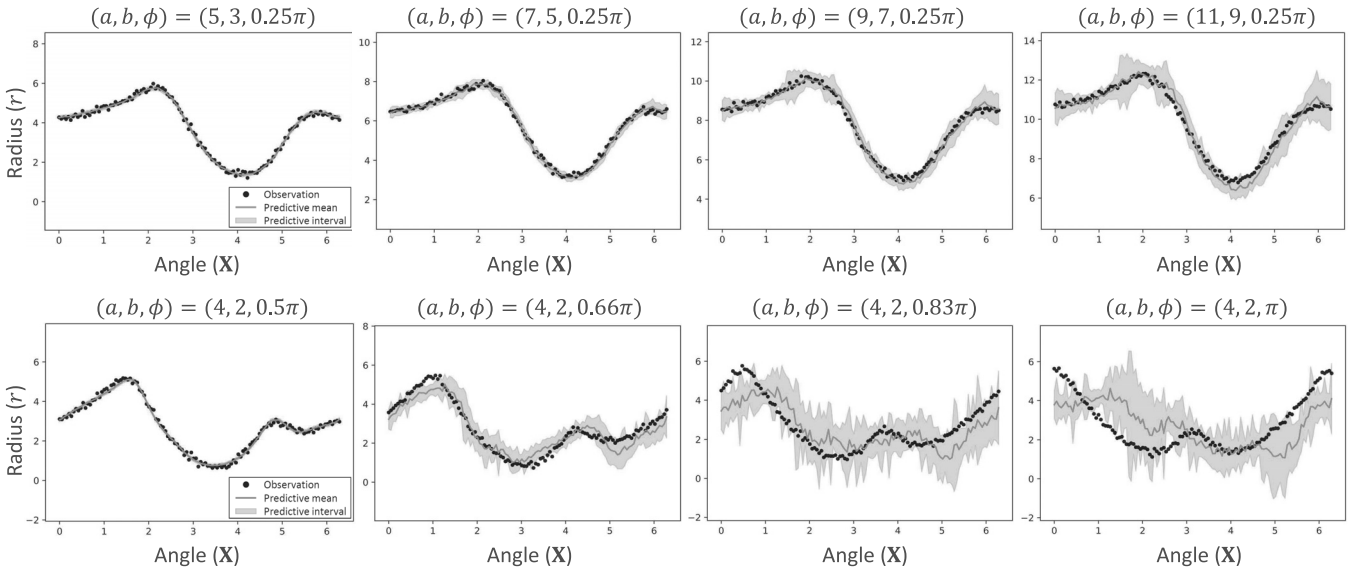

Fig. 9. Predictive means and variances of Multi-BNNs with inverse transformation. Figures in the first row illustrate the cases where the rotation ϕ is fixed whereas in the second row the scaling is fixed. Note that the scale of the y-axis in each row is fixed for the ease of visual comparison while the range might be different.

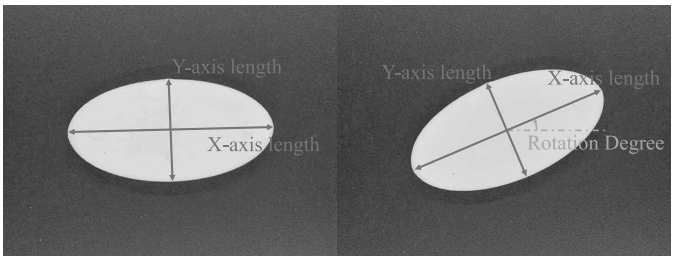

Fig. 10. The ellipse and the dimension terms for the experimental case study: (left) normal printing orientation, (right) printed in a different printing orientation.

A. Case Study I: Elliptical Disks Production

1) Problem Description: The first case study focuses on the production of elliptical disks and aims to quantify geometry or dimension error. We 3D print several ellipses of different sizes, elongation, and printing orientation using an *Ultimaker 3 Extended* 3D printer. A total of 64 representative samples are selected so that all possible combinations of the nominal x-axis length among 20, 40, 60, or 80 mm, nominal y-axis length among 20, 40, 60, or 80 mm, and nominal rotation degree among $-30°$, $-15°$, $0°$, $15°$, or $30°$ are included. The above dimension terms are explained in Fig. 10. We limit the rotation degree between $-45°$ and $45°$. The outputs of interest are the dimension error (from the nominal shape), specifically along the x-axis and y-axis in the ellipse frame. We focus on those two axes because the errors on the axes are the most significant over the other angles, and the dimensions with the extreme length matter most in practice. This output is measured using a *Starrett* electronic Vernier caliper with a measuring resolution of 0.01 mm. Also, four markers are printed above the ellipse to indicate the major and minor axes. The raw data of the measured error is presented in Fig. 11. According to the above experiment design, since ellipses can be transformed from circles by scaling along the x-axis and y-axis and then rotation,

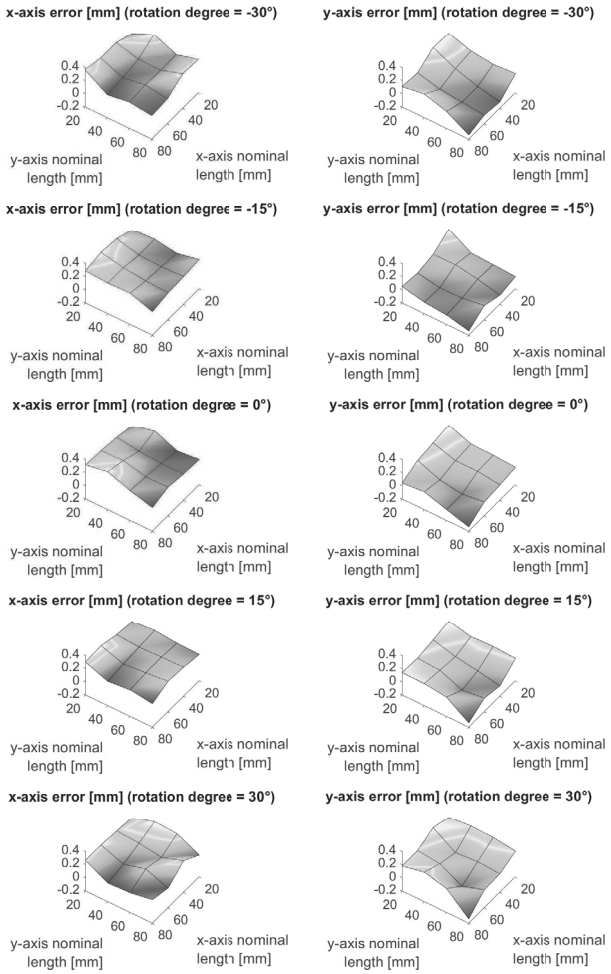

Fig. 11. Surface plots of the measured error (raw data).

the design of the multi-stage approach also follows the same sequence of the imaginary operation. Therefore, the case study

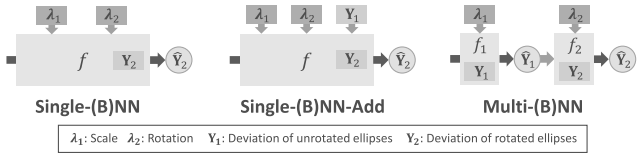

Fig. 12. Inputs and outputs for comparative models in the case study.

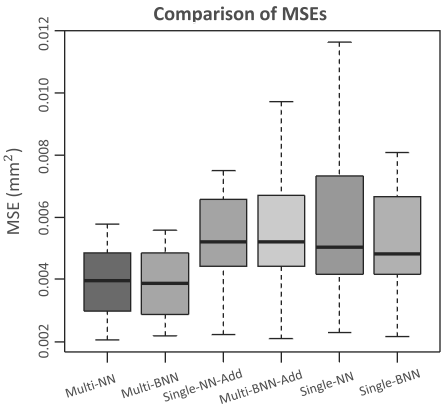

Fig. 13. MSEs of comparative models from the case study using 3D printer dataset.

provides a valid example of Eq. (2) and (3). Multi-(B)NN (i.e., Eq. (3)) performs the first stage to model and predict the scaling effects using the observations with 0° rotation, and then the second stage for the rotation effects. On the contrary, Single-(B)NN-Add (i.e., Eq. (2)) throws all data including 0° rotation into a single-stage model. The inputs and outputs for the comparative models are compared in Fig. 12.

2) Data Generation: We first form a dataset for the multi-stage model as follows. For the first stage, we randomly select 75% of the observations with rotation degree 0° as the training data and remainder as the test data. For the second stage, we form the training data from data with non-zero rotation scaling, but with scaling factors similar to those in the first stage. For the Single-(B)NN-Add we aggregate the training data in the first and second stages. Hence, the comparative models, Multi-(B)NN and Single-(B)NN-Add, are given the same training and test datasets.

3) Results: Fig. 13 provides boxplots summarizing the prediction accuracies of the comparative approaches. Many key insights can be derived. (i) As observed in previous cases, Multi-(B)NN significantly outperforms Single-(B)NN-Add, with around a 17.9% improvement in prediction accuracy. Indeed, this confirms the benefits of knowledge guidance: the deviation of a rotated shape is highly related to that of the unrotated shape, therefore through a simpler model that predicts deviation of the basic unrotated shape one may provide outputs with higher predictive power at the last and final stage. (ii) It is also crucial to note that Single-(B)NN-Add do not provide better predictions than Single-(B)NN even though Single-(B)NN-Add are trained with more observations (non-rotated ellipses). This highlights our initial motivation that blindly throwing more data in highly complex models such as NN, does not always guarantee a better prediction, and hence emphasizes the importance of leveraging

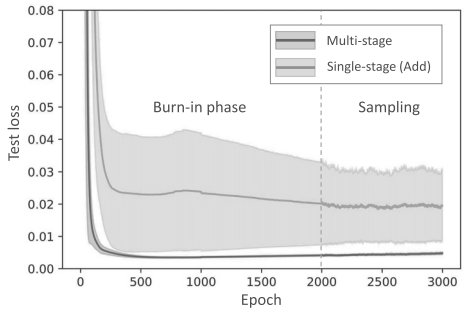

Fig. 14. The trend of test loss in epochs. We record the test losses from 20 evaluations with randomly initialized NNs while we use the same dataset. Solid lines and shaded areas present mean and a half standard deviation of the test loss at each epoch, respectively.

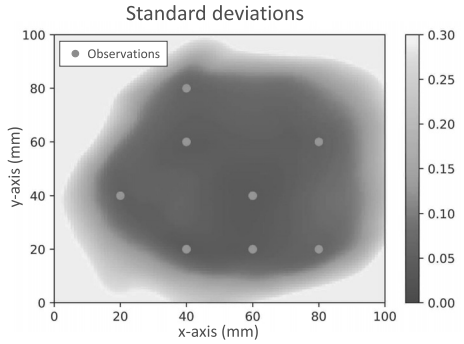

Fig. 15. Heatmap of predictive standard deviations for input space with $\lambda_2 = 15°$.

process-specific knowledge. (iii) BNNs in both single and multi-stage models give slightly less MSEs than corresponding regular NNs on average. The difference however is not significant. We believe this is due to the small sample size. (iv) Lastly, we find that multi-stage models provide robust predictions with lower variance than single-stage models. This is well illustrated in Fig. 14 where we draw the trend of test losses in epochs from 20 repetitions with randomly initialized NNs using the same dataset. The trend shows that the single-stage model gives highly variable test loss depending on the starting point of the optimization. This again confirms the benefit of the proposed model that reduces search space by the decomposition, allowing us to consistently find good local minima.

In Fig. 15 we plot the heatmap of the predictive standard deviation by a BNN in the input space where λ_2 is fixed to 15°. We find that the estimated uncertainty is small for the region in which observations are collected (red dots). Furthermore, the estimated uncertainty at the input $(20, 20)$ or $(20, 60)$ is larger than that of the input $(40, 40)$, as $(40, 40)$ has more adjacent training points around.

B. Case Study II: Stereolithography

1) Problem Description: The second case study aims to learn the dimensional changes of 3D-printed parts throughout printing and post-processing processes. In particular, we consider SLA 3D printing [6] that uses a laser to photopolymerize liquid resin into hardened plastic to form 3D objects. An SLA 3D-printing process comprises three sequential operations:

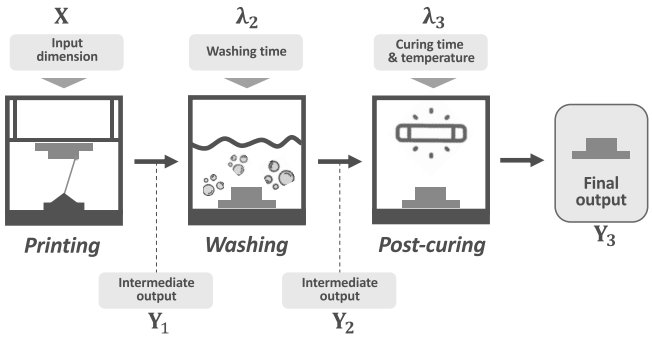

Fig. 16. Inputs and outputs of the sequential operations in SLA additive manufacturing.

printing, washing, and curing, written in order. In the printing process, a 3D printer produces a part using given dimension information. Then the washing process removes liquid resin on the part's surface, followed by the post-curing process that completes the polymerization reaction. One crucial fact is that the operations sequentially make a subtle deformation in printed parts, inducing deviations between the actual dimension of printed parts and the input dimension. Attaining an accurate prediction of such deviations is essential in quality control for 3D printed parts. To this end, we can use the proposed framework to build sequential predictive models where each model learns dimensional changes by each operation in the SLA additive manufacturing process. Our specific focus is on circular 3D-printed disks. We specifically aim to predict the absolute deviation of the maximum disk diameter from the nominal diameter of the circular disk. Figure 16 illustrates inputs and outputs of the SLA process.

2) Data Generation: We printed 70 circular disks with nominal diameters ranging from 14-22mm, using *Form 3*, a Formlab's SLA printer [41]. After printing, we performed washing using Formlab's *FormWash* and then post-curing of the printed disks, using Formlab's *FormCure*. The parameter of the washing process is washing time, set to 5, 10, or 15 minutes. The parameters of the post-curing process are curing time and temperature, set to 30 or 60 minutes and 45, 60, or 75 °C, respectively. Once each process ended, we measured the maximum diameter of each disk using a *Mitutoyo 293-340-30* digital micrometer to get intermediate and final observations. We have made the dataset publicly available[1] [42].

3) Model Training and Evaluation: Each NN at the first, second, and third stage in Multi-(B)NN sequentially learns the maximum diameter deviations after the printing, washing, and post-curing processes, respectively. The NNs take the deviation from the previous process (i.e., an intermediate output) and parameters of the current process to predict the deviation after executing the current process. On the other hand, Single-(B)NN-Add takes the entire data that encompasses all intermediate outputs and process parameters to predict final deviations, while Single-(B)NN only uses process parameters to perform predictions. Predictive models are trained using the observations of 56 randomly selected disks and tested on the rest, repeated 10 times. We evaluate the models based on MSEs of their predictions.

[1]https://doi.org/10.5281/zenodo.5966323

TABLE II
SUMMARY OF THE RESULTS ON CASE STUDY II

Unit: mm^2	M-NN	M-BNN	S-NN-A	S-BNN-A	S-NN	S-BNN
Avg.	**12.33**	12.43	16.58	15.23	18.07	17.80
Std.	2.91	2.71	4.65	6.59	7.95	7.76
Time (sec.)	25.56	33.04	10.41	13.20	7.78	10.55

*M-(B)NN: Multi-(B)NN; S-(B)NN-A: Single-(B)NN-Add; S-(B)NN: Single-(B)NN. **Results are scaled by $\times 10^5$. ***Best results are boldfaced.

4) Results: Table II summarizes the average MSE and its standard deviation across all models.

We first see that the proposed multi-(B)NNs provide more precise predictions compared to the single-stage models. This demonstrates that exploiting knowledge on the operational sequence helps establish an improved predictive analytics tool for an additive manufacturing process with sequential sub-operations. Another advantage of the proposed model is consistency in performance, shown by the lower standard deviation of MSEs. Similar to the first case study, this is attributed to the decomposition that reduces the search space in parameter estimation. Meanwhile, the Bayesian treatment to NNs consistently improves the generalization power of the single-stage models while not requiring significant extra time for inference.

VI. CONCLUSION

This article proposes a knowledge-guided multi-stage inference framework for systems comprised of a sequence of sub-operations. Our approach exploits prior knowledge of the operational sequence to decompose the model into easier sub-problems that are inductively defined. We then propose a Bayesian approach to quantify and propagate uncertainty across operations. Our model can readily incorporate physical knowledge of the output of a sub-operation which is often more practical in reality relative to understanding the physics of the entire process. Our method is then tested on both simulated and real-world data from 3D printed disks.

One possible limitation of our approach is error propagation across the stages. A predictive model can make an inaccurate prediction in real-world applications because the data is highly contaminated by measurement noise or the model converges to poor minima. Such poor predictions that occur at an intermediate stage can quickly propagate to the subsequent stages. Our uncertainty propagation approach is intended to detect such an issue; yet further exploiting diagnostics for early detection of error propagation is an exciting research direction to investigate.

APPENDIX A

Preconditioned SGLD in the Multi-Stage Framework

One challenge in SGLD is that the step size for every weight parameter at each iteration are the same [43], [44]. Because local curvature of the loss surface in BNN is often highly variable in the different directions in the parameter space, moving with the same step size for every parameter results in slow convergence. The preconditioned SGLD [40] thus instead

uses adaptive step sizes. It borrows the idea of an adaptive optimizer called RMSprop [45], introducing a preconditioning matrix to the parameter updating.

Specifically, let $G(\boldsymbol{\theta})$ denote the preconditioner matrix at $\boldsymbol{\theta}$, the parameter update of the preconditioned SGLD in the m-th model is given by

$$\Delta\boldsymbol{\theta}_m^{(t)} = \frac{\eta_t}{2}\Bigg[G(\boldsymbol{\theta}_m^{(t)})\Bigg(\nabla \log P(\boldsymbol{\theta}_m^{(t)})\frac{N}{n}\sum_{i=1}^{n}\nabla \log P(\mathcal{D}_m^{(t,i)}|\boldsymbol{\theta}_m^{(t)})\Bigg) + \Gamma(\boldsymbol{\theta}_m^{(t)})\Bigg] + \tau G^{\frac{1}{2}}(\boldsymbol{\theta}_m^{(t)})\epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0, \eta_t \mathbf{I}), \tag{12}$$

where $\Gamma(\boldsymbol{\theta}) = \sum_j \frac{\partial G_{i,j}(\boldsymbol{\theta})}{\partial \theta_j}$. Also, the preconditioner is updated by

$$G(\boldsymbol{\theta}_m^{(t+1)}) = \mathrm{diag}\Bigg(\mathbf{1} \oslash \Bigg(\alpha\mathbf{1} + \sqrt{V(\boldsymbol{\theta}_m^{(t+1)})}\Bigg)\Bigg)$$

$$V(\boldsymbol{\theta}_m^{(t+1)}) = \beta V(\boldsymbol{\theta}^{(t)}) + (1-\beta)\bar{g}(\boldsymbol{\theta}_m^{(t)}; \mathcal{D}^{(t)}) \odot \bar{g}(\boldsymbol{\theta}_m^{(t)}; \mathcal{D}^{(t)})$$

where $\bar{g}(\boldsymbol{\theta}_m^{(t)}; \mathcal{D}_m^{(t)}) = \sum_{i=1}^{n}\nabla \log P(\mathcal{D}_m^{(t,i)}|\boldsymbol{\theta}_m^{(t)})$ is the sample mean of the gradient of the log likelihood with a minibatch at iteration t; $\alpha > 0$ is a hyperparameter that determines the extremes of the curvature; $\beta \in [0, 1]$ is the hyperparameter for weighted averaging between historical and current gradients; $\odot$ and $\oslash$ are the operators indicating element-wise product and division, respectively.

Comparing to the SGLD update (8), the preconditioned SGLD update (12) features the preconditioning matrix $G(\cdot)$. This is for an adaptation of step sizes over parameters: making step size smaller for the parameters with flat direction, or larger for steep direction. As a result, it converges faster than the SGLD in BNN. Detailed discussions and theoretical analysis are provided in [40].

APPENDIX B

Implementation Settings

A. Simulation Study 1

For Multi-(B)NN, we use a NN with one hidden layer with 30 hidden units for each stage. For Single-(B)NN and Single-(B)NN-Add, we utilize a NN with one hidden layer with 150 hidden units. As a result, the single-stage models are more or similarly flexible to the proposed model. For all NNs, RMSprop optimizer is employed with 2000 iterations. For all BNNs, we place $\mathcal{N}(\mathbf{0}, \mathbf{I})$ as a prior on the weight parameters and use the preconditioned SGLD with total iterations $T = 3000$ including burn-in steps $T' = 2000$ for inference. We set a precondition decay rate $\beta = 0.99$ and thermal noise $\tau = 0.02$. For all comparative models, we schedule decaying learning rates starting from 0.001 and decreasing every 500 epochs by the factor of 0.7.

B. Simulation Study 2

We establish 2 hidden layers with 10 units for NNs in Multi-(B)NN while 20 units in the single-stage models. We also remove the input $\mathbf{X}$ as it appears with every observation and thus is redundant. In optimization, we utilize the RMSprop optimizer with 1000 iterations for the regular NNs and the preconditioned SGLD with 2000 iterations in total where the first 1000 iterations are used for the burn-in steps. We use the same learning rate schedule and detailed hyperparameters for the preconditioned SGLD as in Appendix VI-A.

C. Case Study I

For the multi-stage models, we use a NN that comprises two hidden layers where each layer contains 30 units for the first NN, and 5 units for the second NN. For the single-stage models, we use a NN that comprises two hidden layers where each layer contains 35 units. We employ the NN with more units at the first stage to prevent potential error propagation caused in the first stage. For BNNs, we place a standard multivariate Gaussian distribution $\mathcal{N}(\mathbf{0}, \mathbf{I})$ as a prior. We set burn-in iterations to 10000 and 2000 for the first and second BNNs, respectively, and use 1000 iterations after the burn-in stages for sampling in both models. The hyperparameters of the first and second BNNs are set to $\beta_1 = \beta_2 = 0.99$ (precondition decay rates) and $\tau_1 = 0.0002$, $\tau_2 = 0.002$ (thermal noises). We set a constant learning rate 0.001. We confirmed that every model reached a stable training loss for all optimization processes.

D. Case Study II

For the multi-stage models, we use a NN that comprises two hidden layers. Each NN at the first, second, and third stages has 15, 15, and 20 units for each layer, respectively. For the single-stage models, we use a NN with two hidden layers with 50 units. By doing so, multi- and single-stage models are given similar flexibility. For all BNNs, a prior distribution is set to a standard multivariate Gaussian distribution $\mathcal{N}(\mathbf{0}, \mathbf{I})$. All BNN inferences involve 1500 burn-in iterations, followed by 500 iterations for the sampling stage. Regarding hyperparameters, we set the precondition decay rate β and the thermal noise τ to 0.99 and 0.0005 for all BNNs, respectively. The learning rate of all model training is set to 0.001. We confirmed that every model reached a stable training loss for all optimization processes.

REFERENCES

[1] S. Langarica, C. Ruffelmacher, and F. Nunez, "An industrial internet application for real-time fault diagnosis in industrial motors," *IEEE Trans. Autom. Sci. Eng.*, vol. 17, no. 1, pp. 284–295, Jan. 2020.

[2] R. Zhao, D. Wang, R. Yan, K. Mao, F. Shen, and J. Wang, "Machine health monitoring using local feature-based gated recurrent unit networks," *IEEE Trans. Ind. Electron.*, vol. 65, no. 2, pp. 1539–1548, Feb. 2018.

[3] S. Russel and P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd ed. New York, NY, USA: Pearson, 2009.

[4] P. Xu, F. Roosta, and M. W. Mahoney, "Second-order optimization for non-convex machine learning: An empirical study," in *Proc. SIAM Int. Conf. Data Mining*. Philadelphia, PA, USA: SIAM, 2020, pp. 199–207.

[5] A. Karpatne *et al.*, "Theory-guided data science: A new paradigm for scientific discovery from data," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 10, pp. 2318–2331, Jun. 2017.

[6] F. P. W. Melchels, J. Feijen, and D. W. Grijpma, "A review on stereolithography and its applications in biomedical engineering," *Biomaterials*, vol. 31, no. 24, pp. 6121–6130, Aug. 2010.

[7] A. Mostafaei *et al.*, "Binder jet 3D printing—Process parameters, materials, properties, modeling, and challenges," *Prog. Mater. Sci.*, vol. 119, Jun. 2021, Art. no. 100707.

[8] V.-T. Hoang and K.-H. Jo, "3-D human pose estimation using cascade of multiple neural networks," *IEEE Trans. Ind. Informat.*, vol. 15, no. 4, pp. 2064–2072, Apr. 2019.
[9] Z.-T. Liu, S.-H. Li, M. Wu, W.-H. Cao, M. Hao, and L.-B. Xian, "Eye localization based on weight binarization cascade convolution neural network," *Neurocomputing*, vol. 378, pp. 45–53, Feb. 2020.
[10] H. P. N. Nagarajan *et al.*, "Knowledge-based design of artificial neural network topology for additive manufacturing process modeling: A new approach and case study for fused deposition modeling," *J. Mech. Des.*, vol. 141, no. 2, Feb. 2019, Art. no. 021705.
[11] C. Nemeth and P. Fearnhead, "Stochastic gradient Markov chain Monte Carlo," *J. Amer. Stat. Assoc.*, vol. 116, no. 533, pp. 433–450, 2020.
[12] F. Schrodt *et al.*, "BHPMF—A hierarchical Bayesian approach to gap-filling and trait prediction for macroecology and functional biogeography," *Global Ecol. Biogeography*, vol. 24, no. 12, pp. 1510–1521, Dec. 2015.
[13] K. C. Wong, L. Wang, and P. Shi, "Active model with orthotropic hyperelastic material for cardiac image analysis," in *Proc. Int. Conf. Funct. Imag. Modeling Heart*. Cham, Switzerland: Springer, 2009, pp. 229–238.
[14] J. Xu, J. L. Sapp, A. R. Dehaghani, F. Gao, M. Horacek, and L. Wang, "Robust transmural electrophysiological imaging: Integrating sparse and dynamic physiological models into ecg-based inference," in *Proc. Int. Conf. Med. Image Comput. Comput.-Assist. Intervent*. Cham, Switzerland: Springer, 2015, pp. 519–527.
[15] H. Denli *et al.*, "Multi-scale graphical models for spatio-temporal processes," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 27, 2014, pp. 316–324.
[16] S. Chatterjee, K. Steinhaeuser, A. Banerjee, S. Chatterjee, and A. Ganguly, "Sparse group lasso: Consistency and climate applications," in *Proc. SIAM Int. Conf. Data Mining*, Apr. 2012, pp. 47–58.
[17] J. Liu, K. Wang, S. Ma, and J. Huang, "Accounting for linkage disequilibrium in genome-wide association studies: A penalized regression method," *Statist. Interface*, vol. 6, no. 1, p. 99, 2013.
[18] Z. Geng and Y. Wang, "Physics-guided deep learning for predicting geological drilling risk of wellbore instability using seismic attributes data," *Eng. Geol.*, vol. 279, Dec. 2020, Art. no. 105857.
[19] R. Zhang, Y. Liu, and H. Sun, "Physics-guided convolutional neural network (PhyCNN) for data-driven seismic response modeling," *Eng. Struct.*, vol. 215, Jul. 2020, Art. no. 110704.
[20] X. Jia *et al.*, "Physics-guided machine learning for scientific discovery: An application in simulating lake temperature profiles," *ACM/IMS Trans. Data Sci.*, vol. 2, no. 3, pp. 1–26, May 2021.
[21] J. Shi, *Stream of Variation Modeling and Analysis for Multistage Manufacturing Processes*. Boca Raton, FL, USA: CRC Press, 2006.
[22] J. Jin and J. Shi, "State space modeling of sheet metal assembly for dimensional control," *J. Manuf. Sci. Eng.*, vol. 121, no. 4, pp. 756–762, 1999.
[23] S. Zhou, Q. Huang, and J. Shi, "State space modeling of dimensional variation propagation in multistage machining process using differential motion vectors," *IEEE Trans. Robot. Autom.*, vol. 19, no. 2, pp. 296–309, Apr. 2003.
[24] Q. Huang, J. Shi, and J. Yuan, "Part dimensional error and its propagation modeling in multi-operational machining processes," *J. Manuf. Sci. Eng.*, vol. 125, no. 2, pp. 255–262, 2003.
[25] Y. Gal and Z. Ghahramani, "Dropout as a Bayesian approximation: Representing model uncertainty in deep learning," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1050–1059.
[26] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané, "Concrete problems in AI safety," 2016, *arXiv:1606.06565*.
[27] M. Abdar *et al.*, "A review of uncertainty quantification in deep learning: Techniques, applications and challenges," 2020, *arXiv:2011.06225*.
[28] A. Kristiadi, M. Hein, and P. Hennig, "Being bayesian, even just a bit, fixes overconfidence in Relu networks," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 5436–5446.
[29] D. J. C. MacKay, "A practical Bayesian framework for backpropagation networks," *Neural Comput.*, vol. 4, no. 3, pp. 448–472, May 1992.
[30] R. M. Neal, *Bayesian Learning for Neural Networks*, vol. 118. Cham, Switzerland: Springer, 2012.
[31] A. G. Wilson and P. Izmailov, "Bayesian deep learning and a probabilistic perspective of generalization," 2020, *arXiv:2002.08791*.
[32] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, "Variational inference: A review for statisticians," *J. Amer. Stat. Assoc.*, vol. 112, no. 518, pp. 859–877, 2017.
[33] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural network," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 1613–1622.
[34] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," in *Proc. Int. Conf. Learn. Represent.*, 2014, pp. 1–14.
[35] M. D. Hoffman, D. M. Blei, C. Wang, and J. Paisley, "Stochastic variational inference," *J. Mach. Learn. Res.*, vol. 14, no. 5, pp. 1303–1347, 2013.
[36] M. Welling and Y. W. Teh, "Bayesian learning via stochastic gradient Langevin dynamics," in *Proc. 28th Int. Conf. Mach. Learn. (ICML)*, 2011, pp. 681–688.
[37] R. M. Neal *et al.*, "Mcmc using Hamiltonian dynamics," *Handbook Markov Chain Monte Carlo*, vol. 2, no. 11, p. 2, 2011.
[38] F. Wenzel *et al.*, "How good is the Bayes posterior in deep neural networks really?" 2020, *arXiv:2002.02405*.
[39] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.
[40] C. Li, C. Chen, D. Carlson, and L. Carin, "Preconditioned stochastic gradient Langevin dynamics for deep neural networks," in *Proc. AAAI Conf. Artif. Intell.*, vol. 30, no. 1, 2016, pp. 1788–1794.
[41] Formlabs. (2022). *Stereolithography (SLA) 3D Printing Guide*. Accessed: Feb. 3, 2022. [Online]. Available: https://formlabs.com/blog/ultimate-guide-to-stereolithography-sla-3d-printing/
[42] C.-H. Chou and C. Okwudire, "A case study of the dimensional effects in an additive manufacturing process with multiple operations," Zenodo, Feb. 2022, doi: 10.5281/zenodo.5966323.
[43] A. Dubey, S. J. Reddi, B. Póczos, A. J. Smola, E. P. Xing, and S. A. Williamson, "Variance reduction in stochastic gradient Langevin dynamics," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 29, 2016, p. 1154.
[44] W. Li, S. Ahn, and M. Welling, "Scalable MCMC for mixed membership stochastic blockmodels," in *Proc. Artif. Intell. Statist.*, 2016, pp. 723–731.
[45] T. Tieleman and G. Hinton, "Rmsprop," in *COURSERA: Neural Networks for Machine Learning*. Toronto, ON, Canada: Univ. of Toronto, 2012.

Seokhyun Chung received the bachelor's and Master of Science degrees in industrial and management engineering from Korea University. He is currently pursuing the Ph.D. degree with the Department of Industrial and Operations Engineering, University of Michigan. His research focuses on Bayesian predictive analytics on federated systems and the generalization of deep neural networks.

Cheng-Hao Chou received the B.S.E. degree in mechanical engineering from the National Taiwan University and the M.S.E. degree in mechanical engineering from the University of Michigan, where he is currently pursuing the Ph.D. degree with the Department of Mechanical Engineering. His current research focuses on the data-driven methods for controls of additive manufacturing machines.

Xiaozhu Fang received the B.S. degree in mechanical engineering from Shanghai Jiao Tong University and the dual M.S. degree in mechanical engineering/electrical engineering and computer science from the University of Michigan, Ann Arbor. He is currently pursuing the Ph.D. degree with the School of Science and Engineering, The Chinese University of Hong Kong, Shenzhen. His research focuses on system identification and control theory.

Raed Al Kontar received the B.E. degree in civil engineering in 2014, the M.S. degree in statistics in 2017, and the Ph.D. degree in industrial and systems engineering in 2018. He is currently an Assistant Professor with the Department of Industrial and Operations Engineering, University of Michigan. His research focus is on data science using probabilistic models.

Chinedum Okwudire (Member, IEEE) received the Ph.D. degree in mechanical engineering from the University of British Columbia in 2009. He joined the Mechanical Engineering Faculty, University of Michigan, in 2011. Prior to joining Michigan, he was the Mechatronic Systems Optimization Team Leader at DMG MORI USA, based in Davis, CA, USA. His research is focused on exploiting knowledge at the intersection of machine design, control and, more-recently, computer science and to boost the performance of manufacturing automation systems at low cost. He has received a number of awards, including the Career Award from the National Science Foundation, the Young Investigator Award from the International Symposium on Flexible Automation, the Outstanding Young Manufacturing Engineer Award from the Society of Manufacturing Engineers, the Ralph Teetor Educational Award from SAE International, and the Russell Severance Springer Visiting Professorship from UC Berkeley. He has coauthored a number of best paper award winning papers in the areas of control and mechatronics.