UNIVERSITY
OF OSLO

Anders Thoresen Sandnes

Bayesian machine learning for virtual flow metering

Thesis submitted for the degree of Philosophiae Doctor

Department of Mathematics
Faculty of Mathematics and Natural Sciences

Solution Seeker AS

2024

Series of dissertations submitted to the
Faculty of Mathematics and Natural Sciences, University of Oslo
No. 2739

ISSN 1501-7710

Cover: UiO.
Print production: Graphic center, University of Oslo.

To Gina

Summary

Industrial processes are monitored by an array of measurements. The purpose is to ensure safe and optimal operations. Modern data acquisition and storage technologies have made it possible to collect and distribute large quantities of measurement data in real-time. This ever-increasing stream of information will eventually overwhelm a human analyst, and the need for automated data processing and interpretations will eventually emerge. Data interpretation can take many forms. In this thesis, we are interested in a mathematical "expert commentator" that can provide us with an estimate of unobserved states in a system, based on the real-time information we have available.

The states we are primarily interested in are the quantities of gas, oil, and water that flow through a choke valve. Essentially a speedometer for gas and liquids. This by itself is nothing new. Traditional solutions to this problem typically take one of two forms, both resulting in mathematical models that attempt to predict flow rates based on available measurements. One form is grounded in universal laws of physics, which are used to derive equations that describe the systems of interest. The other is to collect a lot of data from a single system and use it to create an empirical description of the system's behavior. These traditional approaches face different challenges. The flow of gas and liquid mixtures is a complex phenomenon, and the resulting physical equations can quickly become intractable. Additionally, the system is continuously changing, which renders an empirical model outdated.

The presented innovation is to learn a *universal empirical model*. The foundation is data collected from several similar choke valves. Because none of these choke valves are identical, this data set contains descriptions of many variations of the same underlying physics. The strategy is to train joint models that benefit from all these data points. This enables the model to learn fundamental aspects of the system in a purely data-driven fashion.

The proposed mathematical model possesses many beneficial properties. For instance, good predictions and robustness toward changes in the system. While the main focus has been flowrate measurements, other applications have also been explored with promising results.

Sammendrag

Industrielle prosesser blir overvåket av mange måleinstrumenter. Formålet er å sikre trygg og best mulig drift. Moderne datainnhenting og lagringssystemer har gjort det mulig å samle og tilgjengeliggjøre store mengder måledata i sanntid. En stadig økende datamengde vil til slutt overvelde en menneskelig analytiker, og behovet for automatisk prosessering og tolkning av data vil melde seg. En tolkning av data kan være så mangt. I denne avhandlingen er vi interessert i en matematisk "ekspertkommentator" som kan gi oss et estimat på umålte tilstander i et system, basert på den sanntidsinformasjonen vi har tilgjengelig.

Tilstanden vi er ute etter er hvor mye gass, olje, og vann som strømmer gjennom en ventil. I praksis et speedometer for gass og væske. Dette er i seg selv ikke noe nytt. Tradisjonelle løsninger på problemet baserer seg på to strategier. Begge metodene resulterer i matematiske modeller som forsøker å predikere strømningsratene basert på de målingene vi har tilgjengelig. Den ene strategien tar utgangspunkt i universelle fysiske lover, og utleder ligninger som beskriver systemene vi er interessert i. Den andre er å samle en betydelig mengde data fra et enkelt system, og bruke disse til å lage en empirisk beskrivelse av systemets oppførsel. Disse strategiene har hver sine utfordringer. Strømning av gass og væske blandinger er et komplekst fenomen, og de resulterende fysiske relasjonene kan fort bli uhåndterlige. Samtidig er systemet i stadig endring, så en empirisk modell vil raskt gå ut på dato.

Nyskapningen vi legger frem er en *universell empirisk modell*. Utgangspunktet er data som er samlet fra mange liknende systemer. Siden ingen av systemene er identiske, så vil disse dataene beskrive mange ulike varianter av den samme underliggende fysikken. Strategien går ut på å trene felles modeller som deler på disse datapunktene. Dette gjør det mulig å lære fundamentale aspekter ved problemet kun ved hjelp av data.

Den foreslåtte matematiske modellen har flere gunstige egenskaper, blant annet gode prediksjoner og robusthet overfor endringer i systemet. Hovedfokuset har vært anvendelser knyttet til estimering av strømning gjennom ventiler, men andre problemstillinger har også blitt utforsket med lovende resultater.

Preface

This thesis is submitted in partial fulfillment of the requirements for the degree of *Philosophiae Doctor* (Ph.D.) at the University of Oslo. The project is a joint cooperation between the University of Oslo and Solution Seeker AS. The research is supported by Solution Seeker AS and The Research Council of Norway. It is part of the *Industrial Ph.D. scheme*, with project number 298355. The research has been conducted at the University of Oslo and at Solution Seekers offices. Associate Professor Odd Kolbjørnsen has been the main supervisor, representing the University of Oslo. CTO of Solution Seeker, Associate Professor Bjarne Grimstad has been the project leader and main co-supervisor representing Solution Seeker. Professor Geir Storvik and CEO of Solution Seeker Vidar Gunnerud (Ph.D.) have also served as co-supervisors.

The thesis is a collection of four papers. The first is a collaboration with another Ph.D. candidate, Mathilde Hotvedt, who was also co-supervised by Bjarne Grimstad. Bjarne Grimstad is the corresponding author of this work, with support from Mathilde Hotvedt, her main supervisor Professor Lars Imsland, Odd Kolbjørnsen, and myself. The other three papers are written in collaboration with Odd Kolbjørnsen and Bjarne Grimstad, with myself as the corresponding author. The first two papers have been published. The third has been submitted to a journal. The fourth paper is not yet submitted. It is less mature and a major revision must be expected.

Acknowledgements

This work stems from research questions posed at my original workplace, Solution Seeker. The particular research direction of data-driven virtual flow meters is anchored in the core of Solution Seekers technology. I am fortunate that a part of this could be framed as an Industrial Ph.D. project, and that the Department of Mathematics at the University of Oslo where willing to participate. The project has been a curious blend of engineering practicalities, theoretical pondering, and notational confusion. I truly believe that we have created something of both theoretical and practical value, and I am excited to explore its potential with my continued work at Solution Seeker.

I owe much to my supervisors. A huge thanks to Odd Kolbjørnsen for accepting the role of the main supervisor for a project with such an obscure description, and guiding it toward an academically sound product. I am also grateful to Bjarne Grimstad, for taking the role of the project leader and masterfully balancing the interests of Solution Seeker, the University of Oslo, and myself. Odd and Bjarne have both been incredibly helpful and patient during these four years. They have followed the project diligently through all

its phases and deserve much honor for the work presented here. Their broad theoretical expertise and domain knowledge have been invaluable to this project. For this, I am very thankful. I am also indebted to Geir Storvik, for believing in the project from our first contact, and providing many inspiring lectures and discussions early in the project. He probably influenced this work more than he knows. Thanks also go to Vidar Gunnerud for pushing me towards a Ph.D. in the first place and providing great reflections on all aspects of this work.

I further wish to thank my amazing colleagues at Solution Seeker, who, directly and indirectly, knowingly or unknowingly, have helped me tremendously throughout these four years. Without them, this would not have been possible.

Finally, I wish to thank my family, for all their support throughout this journey and all other journeys in my life. And most of all, to my fantastic partner Gina, who makes the road so much better.

Anders Thoresen Sandnes
Oslo, March 2024

List of Papers

Paper I

Bjarne Grimstad, Mathilde Hotvedt, Anders T. Sandnes, Odd Kolbjørnsen, Lars S. Imsland, "Bayesian neural networks for virtual flow metering: An empirical study". *Applied Soft Computing* Volume 112, 2021, 107776, ISSN 1568-4946,

Paper II

Anders T. Sandnes, Bjarne Grimstad, Odd Kolbjørnsen, "Multi-task learning for virtual flow metering". *Knowledge-Based Systems* Volume 232, 2021, 107458, ISSN 0950-7051,

Paper III

Anders T. Sandnes, Bjarne Grimstad, Odd Kolbjørnsen, "Multi-task learning by learned context neural networks". Submitted to journal.

Paper IV

Anders T. Sandnes, Bjarne Grimstad, Odd Kolbjørnsen, "Sequential Monte Carlo applied to virtual flow meter calibration". To be submitted.

Contents

Chapter 1

Introduction

Soft sensors are real-time systems that utilize mathematical models in combination with physical instrumentation to predict quantities of interest in industrial assets (Jiang et al., 2021). Soft sensing can be an economical alternative to installing additional physical measurements (Lu et al., 2019), and they can be used to predict quantities that are infeasible to measure by physical devices due to the hostility of the environment or lack of access to the locations of interest.

In recent years, industrial assets have seen a steady increase in data collection, improved data infrastructure, and increased computing power (Isaksson, Harjunkoski, and Sand, 2018), which are essential ingredients in most soft sensing schemes. This, combined with significant advances within statistical and machine learning methodology (Gelman and Vehtari, 2021; Schmidhuber, 2015), has made data-driven soft sensing an active research area (Jiang et al., 2021; Kadlec, Gabrys, and Strandt, 2009; Sun and Ge, 2021).

In this thesis, we explore a particular soft sensing case for oil and gas production networks. The oil and gas industry is still seen as being in the early stages of digitalization. The industry has the data and computing infrastructure but lacks the intelligent systems needed to leverage the large quantities of data being collected daily (Lu et al., 2019). A broad range of data-driven applications has been researched, but the adoption rate is low (Balaji et al., 2018).

Our focus is on virtual flow meters. Flow meters are the speedometers of the oil field. Knowledge of flow rates throughout the asset is essential in modern operations and production planning. Production optimization (Foss, Knudsen, and Grimstad, 2018), reservoir management (Kanshio, 2020), and flow assurance (Jamaluddin and Kabir, 2012) all rest on knowledge about the flow rates from each well. The value of such rates is clear, but they can still be too costly and challenging to acquire (Hansen, Pedersen, and Durdevic, 2019). With decreasing profit margins, there is an increasing interest in the cost-efficient solutions that virtual flow metering promises (Lu et al., 2019). Data-driven virtual flow metering is thus a compelling application.

This project seeks to answer questions related to data-driven virtual flow metering. We are interested in exploring the limitations and possibilities that lie in purely data-driven models. Historically, virtual flow metering has, in practice, been dominated by first principle models, while data-driven methods have been restricted to research applications. We are interested in why this is the case and wish to uncover challenges faced by data-driven methods in practice and attempt to propose a solution to these challenges.

The data at our disposal is *well-tests*, which are pairs of measurements taken locally at the well and corresponding flow rates observed at a test separator.

These are our main sources of information, as it pairs the measured state of the well with the flow rates we wish to predict. A second source of information is *production data*, which measures the joint production of all wells simultaneously. As such, it pairs the measured state from all the wells with their combined flow rates. well-tests are highly informative, but sparsely distributed in time. Production data is less informative but continuously available.

The proposed strategy is comprised of two components. One static and one dynamic. One for well-tests and one for production data. The static component is a model that predicts the well flow rates based on the available measurements and *assumed knowledge* about the flow composition. A *single model* will be shared between all wells, which allows it to learn from all the well-tests. The knowledge assumed by the first component is provided by the second. The second component is a dynamic state-space model that estimates how the flow composition develops over time. To achieve this, it utilizes the continuously available data from the production separator to monitor the collective development of all the producing wells.

1.1 Contributions

The research contributions of this project are presented in four papers. Three of these are directly addressing data-driven virtual flow metering. They attempt to shed light on the topics of model uncertainty, model generalization, and model calibration respectively. Uncertainty is studied with Bayesian neural networks, generalization by multitask learning, and calibration through sequential Monte Carlo. Of these three branches, most research efforts were put into multitask learning, and the fourth paper is dedicated solely to a deep dive into the proposed multitask learning architecture.

The papers are briefly summarized as follows. Paper I is a large-scale study of data-driven virtual flow meters where both conventional and probabilistic neural networks are applied. It concludes with a series of data-related challenges faced by traditional data-driven virtual flow meter strategies. Paper II is a similar study, but with multitask learning being introduced to address the challenges identified in Paper I. The results are favorable toward the proposed architecture. Paper III is a theoretical deep-dive into the multitask neural network architecture introduced in Paper II. It establishes the proposed architecture as a valid alternative to the classical multitask learners and studies its performance on several public data sets from different domains. Paper IV explores the time-varying aspect of virtual flow meter models. It presents a calibration strategy that attempts to keep model performance up to date after the initial construction. The strategy has a quite general state-space formulation at its core, which makes it, almost, agnostic to the virtual flow meter model.

1.2 Outline

The remainder of this thesis is outlined as follows. Chapter 2 provides theoretical background on statistics and machine learning and gives an introduction to the virtual flow metering domain. Chapter 3 provides an outline of the modeling strategy. Chapter 4 explores the data used in the research and illustrates the challenges they present. Chapter 5 elaborates on the publications in light of the data challenges and the overall modeling strategy. Finally, Chapter 6 discusses the research findings and concludes. The papers are provided in full length at the end.

Chapter 2

Background

This chapter introduces methodical and domain background which is the basis for the research. We focus our attention gradually, by first presenting *statistical and machine learning* frameworks for inference Section 2.1. Then a discussion of the the broader theme of *soft sensing* is given in Section 2.2. Finally, the domain specifics of *virtual flow metering* are treated in Section 2.3.

2.1 Statistical and machine learning methodology

The goal of this thesis is to learn how to predict future events from past observations. Our learning methods are rooted in theory from statistical inference and machine learning. Statistical inference draws conclusions about unobserved quantities from the data (Gelman, Carlin, et al., 2013). Machine learning is concerned with computer algorithms that extract knowledge and patterns from data (Goodfellow, Bengio, and Courville, 2016). These domains have many aspects in common but with different historical developments. Therefore we find similar concepts denoted by different names and different concepts that have similar names. In the following, we reconcile notation and theoretical frameworks that give the background for this project, clarifying the statistical and machine learning aspects when required. We start with a brief review of notation and general concepts from statistical modeling. The link between the data-generating process and the models is then given, followed by a description of different statistical models of interest. We then proceed with a discussion on how we can formulate learning problems that yield models with desirable properties. Finally, the actual methods for inference and learning are presented.

The discussions on statistical modeling and inference are primarily based on Hastie, Tibshirani, and Friedman (2009) and Gelman, Carlin, et al. (2013), while Goodfellow, Bengio, and Courville (2016) gives the machine learning aspect. Discussions on computer algorithms are primarily based on Givens and Hoeting (2012) and Nocedal and Wright (2006).

2.1.1 Learning problems

Consider a sequence of n observations, $y_1, y_2, \ldots, y_n$, that are identically distributed with probability density $p(y)$. We wish to use the observations to learn about the process that generated them. Our goal may be to understand some aspect of the process or to predict future observations. For some processes, we have access to additional quantities, $x_1, x_2, \ldots, x_n$, which are paired with our primary observations, $(y_i, x_i),\ i = 1, \ldots, n$. These are identically distributed

according to their *joint* distribution, $p(y, x)$, which can be factorized as

$$p(y, x) = p(y|x)p(x). \tag{2.1}$$

We may wish to limit our study to the *conditional* distribution $p(y|x)$, if the quantities $x_1, \ldots, x_n$ are not considered to be random.

Supervised learning is concerned with the conditional distribution $p(y|x)$, and seeks to learn the relationship between the *explanatory variables*, x, and the *response*, y. In the context of machine learning, these are also referred to as features and labels respectively. The goal is to construct a *model* that can predict a future response given a set of new explanatory variables. The learning problem is called a *regression* problem if the response is quantitative, or continuous, and a *classification* problem if it is qualitative, or discrete (Hastie, Tibshirani, and Friedman, 2009). *Unsupervised learning* is concerned with general patterns in the data, for instance, if it is organized in clusters. It might be interpreted as if there is no response variable in the data (Hastie, Tibshirani, and Friedman, 2009). In Equation (2.1), unsupervised learning techniques may, for instance, be used to study $p(x)$. Supervised and unsupervised learning are well-established concepts in the literature. But there are other emerging paradigms, such as semi-supervised learning (Engelen and Hoos, 2019) and reinforcement learning (Kaelbling, Littman, and Moore, 1996), that are emerging. Our discussion of statistical models is focused on supervised learning and the case of regression models. Identifying the quantities of interest from a given dataset is known as *training* in the machine learning context and *inference* in the statistical context.

2.1.2 Data, model, and likelihood

We consider a dataset of observations

$$\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{n}, \tag{2.2}$$

with vector of explanatory variables $x \in \mathcal{X}$, and response $y \in \mathcal{Y}$. The response can also be a vector, but we consider it a scalar for simplicity. Our observations are identically distributed as presented in Equation (2.1). We consider supervised learning and the special case of regression, and model the conditional distribution, $p(y|x)$, as

$$y = f(x) + e, \tag{2.3}$$

where e is a random variable with $\mathbf{E}[e] = 0$. Our model is then the conditional expected value of the response given the explanatory variables, $f(x) = \mathbf{E}[y|x]$. The additive term e is often referred to as a *noise* or *error* term. In general, the model is a function $f \in \mathcal{F}$, where $\mathcal{F}$ is the *model space*.

We primarily consider models where the predictions are determined by a set of *parameters*, $\theta \in \Theta$, where Θ is the *parameter space*. To emphasize the dependence on the parameters, we rewrite Equation (2.3) as

$$y = f(x; \theta) + e, \tag{2.4}$$

and the conditional density for this model is $p(y|x, \theta)$. If we also need parameters to describe the statistical properties of the error term, e, we can incorporate this along with θ as necessary. The information that the dataset carries about θ is summarized in the *likelihood* function,

$$\mathcal{L}(\theta) = p(\mathcal{D}|\theta), \tag{2.5}$$

which is the conditional density *viewed as a function of* θ. Consider the case where we study the response given the explanatory variables for all possible values of the parameters. Further, it is common to assume that the observations are independent when the parameters are given. This allows for the simplification

$$\mathcal{L}(\theta) = \prod_{i=n}^{n} p(y_i|x_i, \theta), \tag{2.6}$$

by introducing the parameters, θ, into the marginal density in Equation (2.1). The likelihood function plays an important part in model inference, which will be discussed in Section 2.1.8. The likelihood presented in Equation (2.5) and Equation (2.6) are general expressions, and not limited to the special case of regression.

In addition to the model parameters θ, some models also have *hyperparameters*. The hyperparameters may, for instance, control the model space $\mathcal{F}$ or the parameter space Θ. In machine learning, hyperparameters also frequently include parameters of the learning algorithm used for model inference.

2.1.3 Function approximation

One of the main challenges of statistical regression models is to construct a suitable model space for Equation (2.3). Aspects to consider are the properties we require of our model, and how much information we have available from our data. A property of particular interest to us is *universal approximation*, which, informally, means that our model is able to approximate the desired target arbitrarily well (Hastie, Tibshirani, and Friedman, 2009). More formally, let $\mathcal{X}$ be a compact subspace of $\mathbf{R}^d$, and the target a continuous function $f^\star : \mathcal{X} \to \mathbf{R}$. Our model is a universal approximator if for every $\epsilon > 0$ there exists an $f \in \mathcal{F}$ such that

$$\sup_{x \in \mathcal{X}} ||f(x) - f^\star(x)|| < \epsilon. \tag{2.7}$$

The existence of such a model does not mean we are able to identify it. The statement above only considers a scalar response. In the case where the response is multidimensional, we apply the argument for one dimension at a time.

The degree to which a particular model, or class of models, is able to approximate arbitrary functions is sometimes called approximative power. We generally do not explicitly quantify the approximative power of individual models but will use the concept to order different models in terms of being more or less powerful. In the context of models such as Equation (2.4), we generally have that an increasing number of parameters yields a more powerful model.

2.1.4 Regression models

We now proceed by reviewing a selection of statistical models and describing them with a uniform notation. For all the statistical models presented below we have $x \in \mathbf{R}^d$ and $y \in \mathbf{R}$, data as defined in Equation (2.2), and a model as defined by Equation (2.4). While we limit our discussion to regression problems, many of the models considered below can easily be extended to classification problems.

The most commonly applied statistical model is the linear regression model (Gelman, Carlin, et al., 2013). Let $\theta = \{\alpha, \beta\}$, where $\alpha \in \mathbf{R}^d$ and $\beta \in \mathbf{R}$. The linear regression model is a scaled sum of the explanatory variables,

$$f(x;\theta) = \beta + \alpha^\top x \tag{2.8}$$

$$= \beta + \sum_{k=1}^{d} \alpha_k x_k. \tag{2.9}$$

Here we explicitly specify an intercept term, β, but it can also be implicitly included in the explanatory variables, for instance as $x_{d+1} = 1$.

Moving beyond linear models opens up for a variety of function classes, many of which use Equation (2.9) a building block. We first consider basis expansions. Let $\theta \in \mathbf{R}$ and $h_k : \mathbf{R}^d \to \mathbf{R}$ for $k = 1, \ldots, K$. Additionally, collect the parameters and basis function in vectors, $\theta = \begin{bmatrix}\theta_1 & \ldots & \theta_K\end{bmatrix}^\top$ and $h(x) = \begin{bmatrix}h_1(x) & \ldots & h_K(x)\end{bmatrix}^\top$. The basis expansion can then be represented by two equivalent forms,

$$f(x;\theta) = \theta^\top h(x) \tag{2.10}$$

$$= \sum_{k=1}^{K} \theta_k h_k(x). \tag{2.11}$$

The basis expansion is similar to the linear model, except that the explanatory variables are replaced by a set of K new transformed variables. The basis functions can be constructed to span a particular function space, such as the space polynomials or degree K with $h_k(x) = x^k$, or crafted manually through application-specific *feature engineering*. In the case of a generated basis, the number of elements, K, can be seen as a hyperparameter. Once the basis expansion has been fixed, Equation (2.11) becomes a linear regression problem identically to Equation (2.9).

Another class of nonlinear extensions is additive models. Additive models is a sum of contributions from each explanatory variable, as in the linear model, but each variable is subject to arbitrary nonlinear transformation. Given unspecified unspecified $h_k : \mathbf{R} \to \mathbf{R}$, $k = 1, \ldots, d$, and a set of unspecified parameters $\theta = \{\theta_1, \ldots, \theta_d\}$, the additive model is

$$f(x;\theta) = \sum_{k=1}^{d} h_k(x_k;\theta_k). \tag{2.12}$$

While Equation (2.12) allows for significant flexibility in the choice of h_k, it is not a universal approximator. For instance, bi-linear terms such as $x_1 x_2$ cannot be represented. The advantage of this limitation is that the effect of the explanatory variables can be studied individually, which can be a desirable trait.

Projection pursuit takes the additive concept further, by allowing arbitrary projections of all explanatory variables before the nonlinearities are applied. Let $\alpha_k \in \mathbf{R}^d$, $\beta_k \in \mathcal{B}$, and $h_k : \mathbf{R} \to \mathbf{R}$ for $k = 1, \ldots, K$, and $\theta = \{(\alpha_1, \beta_1), \ldots, (\alpha_K, \beta_K)\}$. Projection pursuit is then given by

$$f(x; \theta) = \sum_{k=1}^{K} h_k(\alpha_k^\top x; \beta_k). \tag{2.13}$$

The nonlinearities, h_k, are estimated along with the projections and can take parameters of their own. If we allow for arbitrarily many terms K combined with an appropriate choice of h_k, the projection pursuit model is a universal approximator (Hastie, Tibshirani, and Friedman, 2009).

Shallow neural networks take a similar form, but with the same, predetermined, nonlinearity, $h : \mathbf{R} \to \mathbf{R}$, applied in all K terms,

$$f(x; \theta) = \sum_{k=1}^{K} \gamma_k h(\alpha_k^\top x + \beta_k). \tag{2.14}$$

where $\theta = \{(\alpha_k, \beta_k, \gamma_k)\}_{k=1}^{K}$ with $\gamma_k, \beta_k \in \mathbf{R}$ and $\alpha_k \in \mathbf{R}^d$. For an appropriately chosen nonlinearity, h, and arbitrarily large K, Equation (2.14) is also a universal approximator (Cybenko, 1989). While seemingly similar to projection pursuit, there are key differences between the two. In projection pursuit, the additive terms are constructed one by one, until a stopping criterion is reached, and the nonlinearities can be adapted to the different projections. In neural networks, the type of nonlinearity and the number of additive terms are chosen first, and all the parameters are then learned simultaneously. Neural networks are discussed further in Section 2.1.5.

Tree based methods represent another strategy for generating flexible sets of basis functions that are capable of universal approximation. We only consider regression trees. Recall our model definition, $f : \mathbf{R}^d \to \mathbf{R}$. A regression tree partitions the domain into regions, $R_k \subset \mathbf{R}^d$, $k = 1, \ldots, K$, such that regions are disjoint and the union of all regions covers the entire domain. Let $\alpha_k \in \mathbf{R}$ and $\beta_k \in \mathcal{B}$ for $k = 1, \ldots, K$, and $\theta = \{(\alpha_1, \beta_1), \ldots, (\alpha_K, \beta_K)\}$. Further, let $I(x \in R_k)$ be an indicator function equal to one if the condition is true and zero otherwise, and let the unspecified parameters β_k describe the region R_k to enable the mapping $R_k = R(\beta_k)$. The predictions of a regression tree can then be written

$$f(x; \theta) = \sum_{k=1}^{K} \alpha_k I(x \in R(\beta_k)). \tag{2.15}$$

The splits are generated recursively on one explanatory variable at a time, leading to a decision tree with one region for each leaf node (Hastie, Tibshirani, and Friedman, 2009).

Finally, we consider the mixtures of experts model,

$$f(x;\theta) = \sum_{k=1}^{K} g_k(x;\alpha)h_k(x;\beta), \tag{2.16}$$

where $\theta = \{\alpha, \beta)\}$ with α and β unspecified. The mixtures of experts model is a weighted sum of expert functions, $h_k : \mathbf{R}^d \to \mathbf{R}$, where the weights are given by gate functions, $g_k : \mathbf{R}^d \to [0, 1]$. It is common to restrict the weights to sum to one, $\sum_{k=1}^{K} g_k(x) = 1$. Both the expert functions and gate functions can take arbitrary sets of parameters. The expert functions can take any form, including all the statistical models discussed above (Yuksel, Wilson, and Gader, 2012). For instance, both the expert and the gate functions can be neural networks (Fedus, Zoph, and Shazeer, 2022).

The abovementioned regression models all have their strengths and weaknesses. Choosing an appropriate model for a particular problem is part of *model selection* (Claeskens and Hjort, 2008). Model selection covers many aspects. For instance, choosing between a linear or non-linear regression model, or the number of basis terms in the basis expansion from Equation (2.11). Additionally, it may include *feature selection*, which is to select a subset of the available explanatory variables to use in the actual model, or other dimensionality reduction techniques.

2.1.5 Deep neural networks

Neural networks are a large class of models with a long and winding history (Schmidhuber, 2015). Colloquially, the term neural networks has come to include such a diverse set of computations that it is challenging to define it properly. Originally, the name referred to constructions such as the shallow neural network in Equation (2.14), but the current state of the art has grown far beyond this function form. We use the term *architecture* to mean how a particular neural network is configured to compute the response as a function of the explanatory variables. We limit our discussion to *feedforward neural networks* (Goodfellow, Bengio, and Courville, 2016). For convenience, we sometimes refer to such models only as *networks*.

Feedforward neural networks are functions $f : \mathbf{R}^d \to \mathbf{R}^r$, that are composed of K nonlinear transforms,

$$f(x;\theta) = (g_K \circ g_{k-1} \circ \cdots \circ g_2 \circ g_1)(x), \tag{2.17}$$

where each function, $g_k : \mathbf{R}^{d_k} \to \mathbf{R}^{d_{k+1}}$, in the composition is known as a *layer*. Each layer is a transform that builds on the previous step,

$$z_{k+1} = g_k(z_k;\theta_k). \tag{2.18}$$

The first layer, also known as the *input layer*, g_1, takes $x = z_1$ as its input, making the dimension $d_1 = d$. The last layer, also known as the *output layer*, g_K, produces $f(x;\theta) = z_{K+1}$ as its output, which makes the dimension $d_{K+1} = r$. The other steps, $z_k \in \mathbf{R}^{d_k}, k = 2, \ldots, K$, are intermediate computations that are not directly observed. The layers that produce these unobserved computations are referred to as *hidden* layers.

We refer to the dimensions d_{k+1} as the *size* of the kth layer. A common design choice is to let all hidden layers have the same size. This is then referred to as the *width* of the neural network, and the number of layers, K, is the *depth*. Both the width and depth are considered hyperparameters. We consider a neural network with more than two layers to be a *deep* neural network.

The layers are made up of two components. The first is an affine mapping, which takes a set of parameters, $\theta_k = \{W_k, b_k\}$, consisting of a matrix $W_k \in \mathbf{R}^{d_{k+1} \times d_k}$ and a vector $b_k \in \mathbf{R}^{d_{k+1}}$. The second is a nonlinear *activation function*, $h_k : \mathbf{R} \to \mathbf{R}$. The rectified linear unit,

$$h_k(\cdot) = \max(0, \cdot), \tag{2.19}$$

is a common choice for activation for hidden layers, $k = 1, \ldots, K-1$, while the activation on the last layer, $k = K$, depends on the learning problem (Goodfellow, Bengio, and Courville, 2016). For regression problems, we use the identity function as activation in the last layer. We now specify Equation (2.18),

$$g_k(z_k; \theta_k) = h_k(W_k z_k + b_k). \tag{2.20}$$

The activation function h_k has a scalar domain, and is applied *element wise* to its vector argument.

Feedforward neural networks are flexible models with a potentially large number of parameters. They are universal approximators, granted that the network is sufficiently wide and deep (Kidger and Lyons, 2020). Yet, there are many neural network models that extend the simple feedforward scheme outlined above. These changes may be motivated by a need to reduce the number of parameters without significantly limiting the approximation power of the model or to capture properties known to be present in the problem. An example is *convolutional* neural networks, which assumes and exploits translational invariance of the input-output relationship (LeCun et al., 1989), which is common in, for instance, images. Another example is *recurrent* neural networks, where the input can have arbitrary size and is sequentially processed (Goodfellow, Bengio, and Courville, 2016). These share similarities with state-space models, which we will discuss in Section 2.1.7.

Other changes to the original feedforward architecture can be motivated by a need to make the network easier to train (Balduzzi et al., 2017). An example is residual skip connections (He et al., 2015). Skip connections augment the layer transform in Equation (2.18) by adding the layer input to the output,

$$z_{k+1} = h_k(W_k z_k + b_k) + z_k, \tag{2.21}$$

This requires that the layers involved have the same dimensions, $d_{k+1} = d_k$. Skip connections can be added to some, or all, of the layers in the network. Equation (2.21) illustrates a skip connection that skips over one layer, but the skip connection can span multiple layers.

2.1.6 Repeated measurements and multitask learning

The models discussed so far assume that our data is such that all observations have an identical distribution. Many problems are faced with data that is collected from similar, yet different, subjects. A model may be required to capture aspects these subjects have in common and their uniqueness at the same time.

Consider m subjects, each with a dataset $\mathcal{D}_j = \{(x_{ij}, y_{ij})\}_{i=1}^{n_j}$, where $x_{ij} \in \mathbf{R}^d$, and $y_{ij} \in \mathbf{R}$. Similarly to Equation (2.3), the data is modeled according to

$$y_{ij} = f_j(x_{ij}) + e_{ij}, \tag{2.22}$$

with $\mathbf{E}[e_{ij}] = 0$. Note that we include the indices, i and j, throughout this discussion, in order to highlight which components that vary with the subjects and their observations, and it is implied that the indices take the values $j = 1, \ldots, m$ and $i = 1 \ldots, n_j$.

This type of observed data is referred to as *repeated measurements* in the statistical literature, and the model in Equation (2.22) may be referred to as hierarchical models, multi-level models, or mixed models (Demidenko, 2004; Raudenbush and Bryk, 2002).

In the machine learning context, Equation (2.22) is a case of *multitask learning* (Zhang and Yang, 2021). We use the multitask learning nomenclature in our discussion and refer to the individual learning problems, consisting of a function, $f_j : \mathbf{R}^d \mapsto \mathbf{R}$, and a dataset, $\mathcal{D}_j$, as a *task*. The task functions, $f_j,\ j = 1, \ldots, m$, are assumed to have some aspects in common. We also assume that the tasks are *homogeneous*, which means that the explanatory variables and responses represent the same quantities for all tasks (Zhang and Yang, 2021). If the tasks are treated individually, without any information being shared between them, we refer to it as single task learning. When two or more tasks are trained together in a way that allows information to be shared between them, it is a multitask learning problem.

As an example, consider an industrial plant with multiple installations of the same equipment, and we require a model that predicts the same quantity for all installations. If we assume that the installations are *identical*, we could apply the models from Section 2.1.4 directly. If, instead, the different installations have been exposed to different wear and tear, or they have slightly different, but unknown, properties, we would like to capture these differences to create better predictions for each installation. We would consider this a homogeneous problem if the installations have the same measurements available as explanatory variables and responses. If this is not the case, and they have different measurements or

we wish to predict different quantities as the response, it would be considered a heterogeneous multitask learning problem.

One strategy for dealing with multitask learning problems is to construct models where a subset of the parameters are *shared* among all task and some parameters are *task specific*. The simplest example is the linear *varying intercept* model (Demidenko, 2004). Let $\alpha \in \mathbf{R}^d$ and $\beta_0 \in \mathbf{R}$ be shared parameters, $\beta_j \in \mathbf{R},\ j = 1, \ldots, m$ be task parameters, and $\theta = \{\alpha, \beta_0, \beta_1, \ldots, \beta_m\}$. The varying intercept model is then given by

$$f_j(x_{ij}; \theta) = \alpha^\top x_{ij} + \beta_0 + \beta_j. \tag{2.23}$$

This allows each task to learn its own intercept, while the slope is the same for all of them.

Equation Equation (2.23) can be rewritten to an alternative form. Let $\beta = \begin{bmatrix}\beta_1 & \ldots & \beta_m\end{bmatrix}^\top$ be a vector of the task parameters, $I(k = j),\ k,\ j \in \mathbf{N}$ be an indicator function equal to one if the condition is true and zero otherwise, and

$$c_j = \begin{bmatrix} I(1 = j) \\ \vdots \\ I(m = j) \end{bmatrix}. \tag{2.24}$$

We refer to c_j as the *context vector*, which is a one-hot encoding of the task number. We can now rewrite Equation (2.23) in terms of the context vector,

$$f_j(x_{ij}; \theta) = \alpha^\top x_{ij} + \beta^\top c_j + \beta_0. \tag{2.25}$$

This view presents the model as if the task information is part of the explanatory variables, and hides the individual task parameters to some extent. We can take the concept one step further by defining

$$\tilde{x}_{ij} = \begin{bmatrix} x_{ij} \\ c_j \end{bmatrix} = \begin{bmatrix} x_{ij,1} \\ \vdots \\ x_{ij,d} \\ I(1 = j) \\ \vdots \\ I(m = j) \end{bmatrix}. \tag{2.26}$$

and

$$\tilde{\alpha} = \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_d \\ \beta_1 \\ \vdots \\ \beta_m \end{bmatrix}. \tag{2.27}$$

Which allows us to simplify Equation (2.23) to

$$f_j(x_{ij};\theta) = \tilde{\alpha}^\top \tilde{x}_{ij} + \beta_0. \tag{2.28}$$

With this notation, it is only the indexing that reveals that this model is any different from the ordinary linear regression model in Equation (2.9).

The parameter sharing concept illustrated in Equation (2.23) can be extended to all the nonlinear models discussed in Section 2.1.4. We limit our discussion to multitask neural networks (Zhang and Yang, 2021), focusing our attention on simple augmentation of the feedforward neural networks. Let $h : \mathbf{R}^d \to \mathbf{R}^r$ be a feedforward neural network with $K > 1$ layers, parametrized by $\theta = \{W_k, b_k\}_{k=1}^K$, where the dimensions and computations follow the outline from Section 2.1.5.

The first multitask neural network to consider is the classic architecture of Caruana (1997), which is simply a neural network with *one output for each task*, $h : \mathbf{R}^d \to \mathbf{R}^m$. Upon evaluation of a particular datapoint, x_{ij}, the network simultaneously computes the response of all m tasks, and we select the jth output as the response of interest. In combination with the context vector defined in Equation (2.24), we can write the model of the jth task as

$$f_j(x_{ij};\theta) = c_j^\top h(x_{ij};\theta). \tag{2.29}$$

Let $\tilde{h} : \mathbf{R}^d \to \mathbf{R}^{d_K}$ be the neural network comprised of the first $K-1$ layers of h and $\tilde{\theta}$ be the corresponding parameters. Further, let the last activation function g_K be the identity function. This allows us to write Equation (2.29) as

$$f_j(x_{ij};\theta) = c_j^\top \left(W_K \tilde{h}(x_{ij};\tilde{\theta}) + b_K\right), \tag{2.30}$$

$$= c_j^\top W_K \tilde{h}(x_{ij};\tilde{\theta}) + c_j^\top b_K. \tag{2.31}$$

We can further deconstruct the computation in the last layer. Let $\beta_{j,1} \in \mathbf{R}^{d_K}$ and $\beta_{j,0} \in \mathbf{R}$ for $j = 1, \ldots, m$, such that

$$W_K = \begin{bmatrix} \beta_{1,1}^\top \\ \vdots \\ \beta_{m,1}^\top \end{bmatrix}, \; b_K = \begin{bmatrix} \beta_{1,0} \\ \vdots \\ \beta_{m,0} \end{bmatrix}. \tag{2.32}$$

This is makes $c_j^\top W_K = \beta_{j,1}^\top$ and $c_j^\top b_K = \beta_{j,0}^\top$, and Equation (2.31) can now be rewritten as

$$f_j(x_{ij};\theta) = \beta_{j,1}^\top \tilde{h}(x_{ij};\tilde{\theta}) + \beta_{j,0}. \tag{2.33}$$

This form is identical to the basis expansion in Equation (2.10), where the basis expansion, $\tilde{h}$ is *shared between tasks*. This has the interesting interpretation that a basis expansion is learned from the joint data from all tasks, and that each task performs a linear regression on this basis.

Another strategy is to augment the network *input* to include the context vector c_j (Silver, Poirier, and Currie, 2008). This model is then a neural network

$h : \mathbf{R}^{d+m} \to \mathbf{R}$, where the input is given in Equation (2.26). This is known as a *context sensitive* neural network, and the task models are given by

$$f_j(x_{ij}; \theta) = h(x_{ij}, c_j; \theta). \tag{2.34}$$

This is equivalent to letting each task have its own intercept parameters in the *first* layer of the network. To see this, recall from Equation (2.18), that the first layer is given by $z_2 = g_1(W_1\tilde{x} + b_1)$, where we now focus on the affine mapping,

$$v_{ij} = W_1\tilde{x}_{ij} + b_1. \tag{2.35}$$

Let $\alpha_k \in \mathbf{R}^{d_2},\ k = 1, \ldots, d$, and $\beta_k \in \mathbf{R}^{d_2},\ k = 1, \ldots, m$, be the columns of $W_1 \in \mathbf{R}^{d_2 \times d+m}$,

$$W_1 = \begin{bmatrix} \alpha_1 & \ldots & \alpha_d & \beta_1 & \ldots & \beta_m \end{bmatrix}. \tag{2.36}$$

We now see that Equation (2.35) can be written as

$$v_{ij} = \sum_{k=1}^{d} \alpha_k x_{ij,k} + \sum_{k=1}^{m} \beta_k I(k = j) + b_1. \tag{2.37}$$

Collecting parameters from the first summation,

$$A = \begin{bmatrix} \alpha_1 & \ldots & \alpha_d \end{bmatrix}, \tag{2.38}$$

and noting that there is only one non-zero term in the second summation, we can simplify further

$$v_{ij} = Ax_{ij} + \beta_j + b_1. \tag{2.39}$$

We recognize this structure from the linear varying intercept model from Equation (2.23). In conjunction with the rectified linear unit as the activation function, given in Equation (2.19), this can be interpreted as each task being allowed to adjust its own threshold of activation in the first layer of the network. The adjustments of these changes are then potentially propagated through all the layers of the network to produce the desired adaptation of the response.

In both the classic architecture of Caruana (1997) and the context sensitive architecture of Silver, Poirier, and Currie (2008) there are parameters within the neural network that are implicitly task specific. The number of implicit task parameters is given by the size of the second to last layer, d_K, and the first layer, d_2, respectively. These dimensions can be large, and adjusting them also affects other properties of the network, such as the approximative power of the neural network. The *context adaptation* concept of (Zintgraf et al., 2019) introduces explicit trainable task parameters as input to the neural network. Let $\beta_j \in \mathbf{R}^p$ for $j = 1, \ldots, m$, and the augmented input vector be

$$\tilde{x}_{ij} = \begin{bmatrix} x_{ij} \\ \beta_j \end{bmatrix}. \tag{2.40}$$

Collecting all parameters in $\tilde{\theta} = \{\theta, \beta_1, \ldots, \beta_m\}$, the context adaptation model is then a neural network $h : \mathbf{R}^{d+p} \to \mathbf{R}$,

$$f_j(x_{ij}; \tilde{\theta}) = h(x_{ij}, \beta_j; \theta). \tag{2.41}$$

We refer to this architecture as *learned context* neural networks. To highlight the difference between the learned context and the context sensitive network, we repeat the exercise from Equations (2.35) to (2.39), but with $l_k \in \mathbf{R}^{d_2},\ k = 1, \ldots, p$, as the columns of $L = \begin{bmatrix} l_1 & \ldots & l_p \end{bmatrix}$, we get $W_1 = \begin{bmatrix} A & L \end{bmatrix}$, with A given by Equation (2.38). The affine transform in the first layer of learned context neural networks then becomes

$$v_{ij} = Ax_{ij} + L\beta_j + b_1. \tag{2.42}$$

The columns of L can be interpreted as *latent tasks* and each task is identified as a linear combination of these tasks. In general, L will be a tall matrix, with significantly more rows than columns.

To study the difference between the first layer for learned context and context sensitive networks, given in Equation (2.42) and Equation (2.39) respectively, we study the rank of matrices consisting of the first layer intercepts. Let

$$B^{(cs)} = \begin{bmatrix} \beta_1^{(cs)} & \ldots & \beta_m^{(cs)} \end{bmatrix}, \tag{2.43}$$

be a matrix with columns equal to the context sensitive task intercepts from Equation (2.39). Similarly, for the learned context, we collect the intercepts from Equation (2.42),

$$B^{(lc)} = Lb^{(lc)} = L \begin{bmatrix} \beta_1^{(lc)} & \ldots & \beta_m^{(lc)} \end{bmatrix}. \tag{2.44}$$

We have $B^{(cs)}, B^{(lc)} \in \mathbf{R}^{d_2 \times m}$, $L \in \mathbf{R}^{d_2 \times p}$, and $b^{(lc)} \in \mathbf{R}^{p \times m}$, where d_2 is the size of the first layer, m is the number of tasks, and p is the number of task parameters in the learned context network. We have that $\operatorname{rank}(B^{(cs)}) \leq \min(d_2, m)$ and $\operatorname{rank}(B^{(lc)}) \leq \min(\operatorname{rank}(L), \operatorname{rank}(b^{(lc)})) \leq \min(d_2, m, p)$. Indicating that when $p < m$, the first layer task intercepts for learned context networks may seem like a low-rank variation of the context sensitive equivalent. Additionally, the number of parameters associated with the task intercepts is $d_2 m$ for the context sensitive and $p(d_2 + m)$ for the learned context. For a moderate example with layer size $d_2 = 200$ and $m = 50$ tasks, we get $d_2 m = 10000$, of which each task is responsible for a number of parameters equal to d_2. Compared with $p(d_2 + m) = 250p$ for the learned context, where each task is responsible for only p parameters, which we can freely control to be a small number. This may lead us to believe that the learned context has less flexibility than the context sensitive alternative, but for sufficiently wide and deep neural networks this is not necessarily the case (see Paper III).

Figure 2.1 illustrates the difference between the three multitask neural networks discussed. While deceptively similar, the architectures have some important differences. The difference between context sensitive networks and

learned context networks is that in the learned context case, the entire matrix W_1 is used for all tasks, while in the context sensitive case, parts of the matrix are task specific. Context sensitive networks have implicit task parameters in $\mathbf{R}^{d_2}$, the width of the neural network, while learned context networks have explicit task parameters in $\mathbf{R}^p$. In the classic shared basis architecture, the task parameters enter at the end of the network, which gives them less opportunity to influence the computations. This makes the difference between tasks limited to linear combinations of the same basis, while more complex, nonlinear differences are possible with the input augmentations.

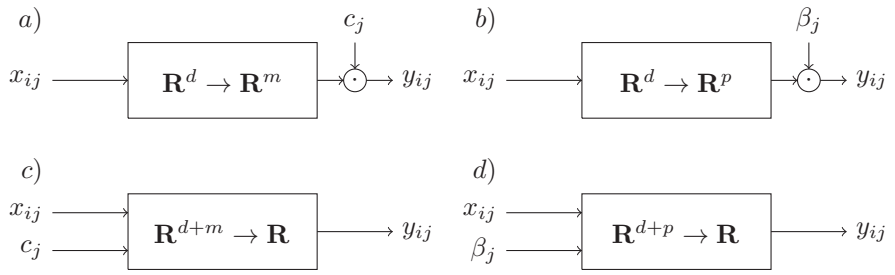


Figure 2.1: Illustration of different MTL neural network strategies. Each box represents a neural network with the dimensions of the domain and co-domain as indicated. The neural networks are shared between tasks. The circles represent the dot product operation. Illustrations *a* and *b* are two different views of *the same* architecture, namely the shared basis neural network of Caruana (1997), given in Equations (2.29) and (2.33). Illustration *c* is the context sensitive neural network of Silver, Poirier, and Currie (2008), given in Equation (2.34). Illustration *d* is the learned context neural network, presented as a meta-learning solution by Zintgraf et al. (2019), given in Equation (2.41). This figure originally appeared in Paper III.

The multitask models we have discussed all make use of *hard* parameter sharing. This means that there are some parameters that are used directly in all task models with the exact same value. An alternative strategy is *soft* parameter sharing. With soft sharing, the parameters are free to take different values for each task, but other mechanisms are introduced to make them similar to each other. This can for instance be done by forcing the parameters to be similar, or by having the resulting computations be correlated with each other S. Yan and X. Yan (2020). Soft parameter sharing can often be formulated with hyperparameters that control the extent of the similarity between tasks, with hard parameter sharing at one extreme and completely independent task parameters at the other.

Part of the motivation for multitask learning is that the task can learn from each other. However, it is not guaranteed that hard, or soft, parameter sharing is beneficial for all tasks. If the model lacks the approximative power to adapt to all tasks simultaneously, then some tasks may see a poorer performance than

if they were treated as a single task learning problem. This is referred to as *negative transfer*.

Multitask learning considers all tasks simultaneously and results in models for all tasks. Other learning strategies propose slight variations on this. *Transfer-learning* is a strategy where the emphasis is on one particular target task, and the data from all other tasks are only used to improve its performance (Hospedales et al., 2022). This can be achieved by first training a model on all the support tasks and then adjusting it with the data from the target task as if it were a single-task learning problem. *Meta-learning* is a strategy where the goal is to find an optimal *learning procedure*. The procedure should be able to learn models for new tasks quickly and with potentially few data points (Hospedales et al., 2022). In practice, these learning paradigms can lead to similar architectures.

2.1.7 State-space models

For both the conventional data and the repeated measurements data, the ordering of the samples is irrelevant, meaning that the order of the observations and tasks can be permuted without any consequences. This is known as exchangeability (Gelman, Carlin, et al., 2013), and is an essential, but often unstated, assumption for many statistical models. There are, however, cases where the observations are not exchangeable and the order of the data is essential, such as when the data is generated by repeatedly sampling from a system that develops over time. This creates a dependency between the observations. State-space models are a class of models designed to address this type of dependency in the likelihood. In these cases we use subscript t instead of i, to emphasize the time dependence. Each time step, t, is associated with an unknown *state*, z_t, and the inference of these states is called *state estimation*.

It is common to assume that the system has the Markov property, which is that the likelihood and transition only depend on the current state. In other words, this means that *all information* about the system up until the current time t, is contained in the current state, z_t, and the history of how we arrived at this state is irrelevant. This simplifies analysis.

Consider a sequence of time steps $t = 1, \ldots, n$, with corresponding unknown states $z_t \in \mathcal{Z}$, response $y_t \in \mathcal{Y}$, and additional explanatory variables $x_t \in \mathcal{X}$. The state-space model takes the form

$$z_{t+1} = g(z_t, x_t) + \epsilon_t, \tag{2.45}$$

$$y_t = f(z_t, x_t) + e_t, \tag{2.46}$$

where e_t and ϵ_t are random variables with expected values equal to zero. Equation (2.45) is known as the transition or state equation, and Equation (2.46) is known as the likelihood or output equation. These equations are considered for all steps, $t = 1, \ldots, n$. The initial state, z_1, is treated differently and may be assumed known or modeled by an appropriate probability distribution.

2.1.8 Maximum likelihood and least squares

Section 2.1.2 introduced the relationship between a statistical model, observed data, and the likelihood function. We now discuss the role of the likelihood function in model inference, compare it to the concept of least squares, which is common in the machine learning literature, and show how the two are related. We limit our discussion to the regression setting, although some statements have a broader applicability.

Recall the setup from Section 2.1.2, with explanatory variables $x \in \mathcal{X}$ and response $y \in \mathcal{Y}$, from which we have observations $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$, that are related by a function $f : \mathcal{X} \to \mathcal{Y}$ taking parameters $\theta \in \Theta$, according to $y = f(x; \theta) + e$, where e is a random variable with $\mathbf{E}[e] = 0$.

The likelihood function $\mathcal{L}(\theta)$, given in Equation (2.5), summarizes all information the data contains about the parameters of our model. We assume that the observations are independent when conditioned on the parameters, which allow us to work with Equation (2.6). The *maximum likelihood estimate* of θ is defined as the parameters that are *the most likely* to have generated the observed data. Formulated in terms of mathematical optimization, we write

$$\arg\max_{\theta \in \Theta} \mathcal{L}(\theta), \tag{2.47}$$

which reads as "find a value of θ, among the candidates in the parameter space Θ, that maximizes the function $\mathcal{L}(\theta)$." Formally, $\theta^\star$ is a solution to Equation (2.47) if $\mathcal{L}(\theta^\star) \geq \mathcal{L}(\theta)$ for all $\theta \in \Theta$.

The *log-likelihood* function, $l(\theta) = \log \mathcal{L}(\theta)$, is equivalent to the likelihood in terms of maximization. This is because the logarithm is strictly monotone, and the two functions therefore take on their maximum value at the same function arguments. With the independence assumption used in Equation (2.6), the log-likelihood is the sum,

$$l(\theta) = \sum_{i=n}^{n} \log p(y_i, x_i | \theta). \tag{2.48}$$

For a common class of statistical models, the log-likelihood can be further simplified, by removing terms and factors that do not influence the solution. Consider the case where $e \sim \mathcal{N}(0, \sigma^2)$. We initially consider σ to be unknown and include along with θ. The conditional density terms are then given by

$$p(y_i | x_i, \theta, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{y_i - f(x_i;\theta)}{\sigma}\right)^2}. \tag{2.49}$$

Inserting this into Equation (2.48) and yields

$$l(\theta, \sigma) = -n \log \sqrt{2\pi} - n \log \sigma - \frac{1}{2\sigma^2} \sum_{i=n}^{n} (y_i - f(x_i; \theta))^2. \tag{2.50}$$

With focus on θ, we note that maximizing the log-likelihood in Equation (2.50) is equivalent to minimizing the *residual sum of squares*,

$$\mathrm{RSS}(\theta) = \sum_{i=1}^{n} \left(y_i - f(x_i;\theta)\right)^2. \tag{2.51}$$

The maximum likelihood estimate is therefore in this case equivalent to the *minimum* residual sum of squares (Hastie, Tibshirani, and Friedman, 2009). The residual sum of squares is also known as the *least squares* estimate. It is the most common regression problem in machine learning, which we will return to in Section 2.1.9.

The linear regression model in Equation (2.9) is well studied in the maximum likelihood setting. We now show its derivation, by solving Equation (2.51), as this is equivalent with respect to the optimal parameters. We do not include an explicit bias term, and let the parameters be defined by $\theta \in \mathbf{d}$, our model be defined by

$$y_i = \theta^\top x_i + e_i, \tag{2.52}$$

$$e_i \sim \mathcal{N}(0, \sigma^2). \tag{2.53}$$

where the observations are

$$\mathcal{D} = \{(y_i, x_i)\}_{i=1}^{n}, \tag{2.54}$$

with $y_i \in \mathbf{R}$, and $x_i \in \mathbf{R}^d$ for $i = 1, \ldots, n$. We collect the observations in vector and matrix form,

$$\mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}, \ \mathbf{X} = \begin{bmatrix} x_1^\top \\ \vdots \\ x_n^\top \end{bmatrix}, \tag{2.55}$$

which allows us to write Equation (2.51) as

$$\mathrm{RSS}(\theta) = (\mathbf{y} - \mathbf{X}\theta)^\top (\mathbf{y} - \mathbf{X}\theta). \tag{2.56}$$

To find the minimum of this expression we differentiate with respect to θ and set this equal to zero,

$$\frac{\partial}{\partial \theta}\mathrm{RSS}(\theta) = -2\mathbf{X}^\top\mathbf{y} + 2\mathbf{X}^\top\mathbf{X}\theta = 0, \tag{2.57}$$

which rearranged yields the solution

$$\hat{\theta} = \left(\mathbf{X}^\top\mathbf{X}\right)^{-1}\mathbf{X}^\top\mathbf{y}. \tag{2.58}$$

The variance of this estimate is

$$\mathrm{Var}[\hat{\theta}] = \left(\mathbf{X}^\top\mathbf{X}\right)^{-1}\sigma^2. \tag{2.59}$$

If the noise variance, σ^2, is unknown, it can be estimated by

$$\hat{\sigma}^2 = \frac{1}{n-d-1}\text{RSS}(\hat{\theta}). \tag{2.60}$$

This is an adjusted version of the maximum likelihood estimator, $\hat{\sigma}^2 = \text{RSS}(\hat{\theta})/n$. The maximum likelihood estimate is biased, and underestimates the variance in this case, because parts of the error terms have, unintentionally, been fitted by the model parameters. In Equation (2.60), the denominator is adjusted to ensure that $\text{E}[\hat{\sigma}^2] = \sigma^2$ (Hastie, Tibshirani, and Friedman, 2009).

Consider now a slight variation of the linear regression problem above, where the noise terms are not identically distributed, but given by $e_i \sim \mathcal{N}(0, \sigma_i^2)$, $i = 1, \ldots, n$, with $\sigma_i > 0$ being known. From Equation (2.50), we are now have a factor $1/\sigma_i^2$ in the least squares loss,

$$WLS(\theta) = \sum_{i=1}^{n} \frac{1}{\sigma_i^2} \left(y_i - f(x_i; \theta)\right)^2, \tag{2.61}$$

which is known as *weighted least squares*. The matrix form of the loss is

$$WLS(\theta) = \left(\mathbf{y} - \mathbf{X}\theta\right)^\top \Sigma^{-1} \left(\mathbf{y} - \mathbf{X}\theta\right), \tag{2.62}$$

where Σ is a diagonal matrix with $\Sigma_{ii} = \sigma_i^2$. Following the same derivation yields the weighted least squares estimate,

$$\hat{\theta} = \left(\mathbf{X}^\top \Sigma^{-1} \mathbf{X}\right)^{-1} \mathbf{X}^\top \Sigma^{-1} \mathbf{y}. \tag{2.63}$$

For the linear regression variations studied above we are able to derive our results analytically. However, for most models, we are unable to do so and must approximate the solutions numerically. This is discussed further in Section 2.1.11.

Under certain regularity conditions, maximum likelihood estimates have desirable limiting properties. This means, that as the number of observations increases without bounds, the estimates will display asymptotic behaviors. One of the key properties is *consistency* (Gelman, Carlin, et al., 2013). Consistency means that if the observations are generated by $p(y|x, \theta^\star)$ and our estimated model is $p(y|x, \hat{\theta})$, then the estimate of $\hat{\theta}$ will converge, in probability, to $\theta^\star$ as the number of observations increases without bounds. Another property is that the estimates are asymptotic normal, which means the distribution of $\hat{\theta}$ converges to a normal distribution with expectation equal to the true value, $\theta^\star$. Of the regularity conditions we highlight *identifiability*, which is that $\theta^\star \neq \theta$ is equivalent to $p(\cdot\ |\theta^\star) \neq p(\cdot\ |\theta)$. This means that different parameters cannot yield "the same" function. A trivial example that violates this is $p(y|x, \alpha, \beta_1, \beta_2, \sigma) = \mathcal{N}(\alpha^\top x + \beta_1 + \beta_2, \sigma^2)$, where two intercept terms, β_1 and β_2, can be varied without affecting the response y as long as they sum to a fixed value. Notably, neural networks are not identifiable.

2.1.9 Loss, regularization, and generalization

Maximum likelihood and minimum residual sum of squares, discussed in Section 2.1.8, are examples of optimization problems that attempt to describe model parameters with particularly desirable properties. In the two cases mentioned, this property is related to having a good fit to the observed data. However, there are other aspects to consider than just the fit to data. In general, we want to combine fit to data with other desirable model properties, in order to identify the best possible model for our application. Formulating such optimization problems and then applying numerical methods to address them is a common strategy for many statistical and machine learning problems.

In terms of optimization, maximizing a function is equivalent to minimizing its negation, and we therefore limit our discussion to minimization problems. In the context of machine learning, we often refer to the function being minimized as the *loss function*, We now discuss the concept of loss functions, and their construction, with more generality. Methods for finding solutions to these optimization problems are discussed in Section 2.1.11.

Let $\theta \in \Theta$ be the parameters of a statistical model, such as the one described by Equation (2.4). Let the loss function be $L : \Theta \to \mathbf{R}$. We write the general minimization problem as

$$\arg\min_{\theta \in \Theta} L(\theta). \tag{2.64}$$

For the maximum likelihood problem we would have $L(\theta) = -\mathcal{L}(\theta)$ from Equation (2.47), and for the residual sum of squares we would take $L(\theta) = \mathrm{RSS}(\theta)$ from Equation (2.51).

The loss functions discussed so far are only concerned with how well the model fits with the data, as they capture the difference between the observed response and model predictions. As discussed in Section 2.1.8, these loss functions display a desirable limiting behavior. However, we will never have infinite data, and a complex model may not be identifiable. Therefore, it is common to augment the loss function with terms that only depend on the parameters, and not the data. This is called parameter *regularization*. Regularization is often introduced to create identifiable problems, find solutions with desirable properties, or to avoid *overfitting* (Hastie, Tibshirani, and Friedman, 2009).

The case of overfitting is particularly relevant in our discussion. Recall the model from Equation (2.4), where the predictions are subject to an additive error term e. If the structure of f is flexible enough, such as the universal approximators discussed in Section 2.1.4, it is possible to find parameters that adapt too well to the specific dataset such that the errors also are fitted. This leads to a model that can have a good fit to the data, but fail to capture the underlying function. This leads to poor *generalization*, which is a model's ability to perform prediction on new, unseen data points.

Regularization is introduced by adding a function $J : \Theta \to \mathbf{R}$ to the loss function. As an example, let $\lambda \geq 0$ and our fit to data be captured by the

residual sum of squares, then the total loss is given by

$$L(\theta) = \text{RSS}(\theta) + \lambda J(\theta), \tag{2.65}$$

where the factor λ determines the balance between the two forces. We can consider λ as a hyperparameter that controls the balance between the model's ability to fit the underlying function and overfitting to noise. Its purpose is to trade between the performance on training data with the ability to generalize to new data. In the case of $\lambda = 0$, Equation (2.65) reduces to the residual sum of squares, and the model attempts to fit the data, including the error terms, as best as possible. As $\lambda \to \infty$, the parameters are forced toward zero. It is natural to expect there to be an optimal value for λ somewhere in between the two extremes. We will return to this question at the end of the section.

In learning problems with many parameters, a common strategy is to force the parameters towards zero. For instance, the squared norm,

$$J(\theta) = ||\theta||_2^2, \tag{2.66}$$

can be used to ensure that no single parameter takes an overly dominating role in the solution. Regularization with the squared norm is known as *ridge regression* when combined with a linear regression model and the least squares loss (Hastie, Tibshirani, and Friedman, 2009).

As an example, consider the the linear regression model from Equations (2.52) to (2.54). We now augmenting the loss function in Equation (2.56) with the squared norm of the parameters,

$$L(\theta) = (\mathbf{y} - \mathbf{X}\theta)^\top (\mathbf{y} - \mathbf{X}\theta) + \lambda \theta^\top \theta, \tag{2.67}$$

and find the solution to the ridge regression problem as

$$\hat{\theta} = \left(\mathbf{X}^\top \mathbf{X} + \lambda I\right)^{-1} \mathbf{X}^\top \mathbf{y}. \tag{2.68}$$

We see that the effect of this regularization is to add a positive diagonal to the matrix being inverted. For an appropriate choice of $\lambda > 0$ this can ensure that the inverse exists even when the original equations are underdetermined. Its effect is that it shrinks the values of θ towards zero as λ increases. The ridge regression solution has an interesting interpretation in terms of the weighted least squares problem from Equation (2.62). If we pretend to have a second set of observations, $0 = \theta_k + e_k$, with $e_k \sim \mathcal{N}(0, \sigma^2/\lambda)$, for $k = 1, \ldots, d$, we can augment the matrix form of the weighted least squares problem to become

$$\mathbf{y}_\star = \begin{bmatrix} \mathbf{y} \\ 0 \end{bmatrix}, \mathbf{X}_\star = \begin{bmatrix} \mathbf{X} \\ I \end{bmatrix}, \Sigma_\star = \begin{bmatrix} \sigma^2 I & 0 \\ 0 & \sigma^2/\lambda I \end{bmatrix}, \tag{2.69}$$

which multiplied out yields an estimate identical to Equation (2.68). This can be interpreted as having observed the values of θ to be zero with the given uncertainty.

A fundamental question is how to quantify a model's ability to generalize to new data, because we naturally do not have access to such data. An underlying assumption is that the new data is drawn from the same distribution as the data we use during model training. A common solution is to simulate the scenario by *splitting* our actually observed data into two disjoint sets. One set is denoted the *training set* and the other the *test set*. The purpose of the test set is to play the role of new data, it is therefore kept untouched during model development. The training data is used to develop and train a model, and its ability to generalization is quantified using the test data (Hastie, Tibshirani, and Friedman, 2009).

Formulating a loss function, such as Equation (2.65), allows us to find the model parameters θ. But, before this can be done, we have to actually specify the model structure and the regularization factor. As an example, consider the basis expansion regression, given in Equation (2.11), with squared norm regularization as in Equation (2.66). We must find suitable values for the number of basis functions, $K \in \mathbf{N}$, and the regularization factor $\lambda \in \mathbf{R}^+$. Ideally, we would want to select these such that the prediction errors on future data are as small as possible. In the machine learning literature, identifying these parameters is referred to as *hyperparameter optimization* (Bergstra and Bengio, 2012).

To make the concept of hyperparameter optimization more precise, let $\phi \in \Phi$ be the hyperparameters of the model and $\theta \in \Theta_\phi$ be the parameters of the model given a particular set of hyperparameters. For the example above, we would have $\phi = (K, \lambda)$ and $\Phi = \mathbf{N} \times \mathbf{R}^+$. The notation Θ_ϕ is to emphasize that the parameter space can change with hyperparameters such as K. It is not trivial to jointly optimize both parameters and hyperparameters, due to the interaction between hyperparameters and the model parameter space. The problem can be seen as a bilevel optimization problem,

$$\arg\min_{\phi \in \Phi} \; L_{\mathrm{v}} \left(\arg\min_{\theta \in \Theta_\phi} \; L_\phi(\theta) \right). \tag{2.70}$$

which consists of an inner and an outer problem (Bengio, 2012). The inner optimization problem is the standard learning problem from Equation (2.64), given a set of hyperparameters, while the outer problem is concerned with the selection of the hyperparameters. The loss function of the inner problem, L_ϕ, may change depending on the hyperparameters, such as in Equation (2.67). The result of the inner optimization is *validated* by the validation loss $L_{\mathrm{v}}(\theta)$. The validation loss is formally defined as

$$L_{\mathrm{v}} : \bigcup_{\phi \in \Phi} \Theta_\phi \to \mathbf{R}, \tag{2.71}$$

meaning that it takes values from the union of all possible parameter spaces generated by the hyperparameters as its argument.

The validation loss is constructed to quantify the different models' ability to generalize, and can be viewed as a model selection criterion. We create the

validation loss by splitting the training data again. One part called *validation data*, is used in the validation loss, while the remaining training data is used for in the inner optimization problem as normal. As an example, let $\mathcal{V} \subset \{1, \ldots, n\}$ be the indices of the data selected as validation data, and θ_ϕ denote the result of the inner optimization problem given a particular set of hyperparameters ϕ. If we choose the residual sum of squares from Equation (2.51) as our validation loss, we may compute it as $L_{\text{v}}(\theta_\phi) = \frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{V}} (y_i - f(x_i; \theta_\phi)^2$.

The hyperparameter optimization can also be susceptible to overfitting. Therefore, once a final set of hyperparameters has been selected, we train a final model on all training data and evaluate this model on the test data set to get a final estimate of the generalization error. Let $\mathcal{T} \subset \{1, \ldots, n\}$ be the indices of the data selected as test data, and recall that the observations used as test data are not used during model training. We may then estimate the variance of the generalization error as the mean residual sum of squares,

$$\hat{\sigma}^2 = \frac{1}{|\mathcal{T}|} \sum_{i \in \mathcal{T}} (y_i - f(x_i; \hat{\theta}))^2, \tag{2.72}$$

where $\hat{\theta}$ is the final parameter estimate. The estimate from Equation (2.72) is unbiased, because the test data is unused during training, and the parameters have therefore not been given the opportunity to fit these error terms. This is in contrast to the maximum likelihood estimator of the noise variance, discussed in Section 2.1.8, which is biased toward underestimating the variance. Equation (2.60) gave a corrected version of the maximum likelihood estimator, making it unbiased by taking the overfitting into account.

The framework discussed above rests on the assumption that future data will be generated from the same distribution as the observations used for training and testing. In some situations, this is not the case. For instance, the explanatory variables may be distributed differently as time passes. This is known as a covariate shift (Quinonero-Candela et al., 2009). Such phenomena make it difficult to quantify the generalization error (Ovadia et al., 2019).

2.1.10 The Bayesian view

We discuss methods to learn from data. We are interested in expressing and updating our belief about some quantity, $\theta \in \Theta$, given the data, $\mathcal{D}$, we have observed. This fits naturally into the *Bayesian* view of inference, which allows us to study the world through the conditional density $p(\theta|\mathcal{D})$. This conditional density is best understood by the factorization

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})}, \tag{2.73}$$

which is the famous *Bayes' rule* (Gelman, Carlin, et al., 2013). We recognize $p(\mathcal{D}|\theta)$ as the likelihood from Equation (2.5). The next term is the *prior* belief, $p(\theta)$, which captures our knowledge *before* any data has been provided. The final term is the marginal data probability, $p(\mathcal{D})$. This is often omitted in

practice when it is sufficient and convenient to study the unnormalized posterior distribution

$$p(\theta|\mathcal{D}) \propto p(\mathcal{D}|\theta)p(\theta). \tag{2.74}$$

With the data as defined in Equation (2.2) and the independence assumptions that led to the likelihood function in Equation (2.6), we can factorize Equation (2.74) as

$$p(\theta|\mathcal{D}) \propto p(\theta) \prod_{i=n}^{n} p(y_i|x_i, \theta). \tag{2.75}$$

A study of the posterior distribution can take many forms. The choice of methodology depends on our needs and computational feasibility.

Consider the regression model as described by Equation (2.4). We may wish to use our model to predict a new response y_0 given a new observation of the explanatory variables x_0. In this case, we have two unknown variables, the parameters and the new data point. We can augment the posterior distribution

$$p(y_0, \theta|x_0, \mathcal{D}) = p(y_0|x_0, \theta, \mathcal{D})p(\theta|x_0, \mathcal{D}), \tag{2.76}$$

for which we can integrate out the parameters to get the *posterior predictive distribution*, $p(y_0|x_0, \mathcal{D})$. A common assumption is that the new data point is independent of the past observations when conditioned on the parameters, which gives the convenient form

$$p(y_0|x_0, \mathcal{D}) = \int p(y_0|x_0, \theta)p(\theta|\mathcal{D})d\theta. \tag{2.77}$$

Analytical posterior densities can be derived for several combinations of likelihoods and priors (Gelman, Carlin, et al., 2013). For instance, revisit the linear regression model from Equations (2.52) to (2.54). We now extend this model with a *prior distribution* on the parameters,

$$\theta \sim \mathcal{N}(\mu, \sigma^2/\lambda I), \tag{2.78}$$

where $\lambda > 0$ is a scalar factor that represents the difference in uncertainty between the prior and the observations. In this case, the posterior distribution then takes the same form as the prior,

$$\theta|\mathcal{D} \sim \mathcal{N}(\hat{\theta}, V_\theta), \tag{2.79}$$

with mean and covariance given by

$$\hat{\theta} = \left(\mathbf{X}^\top \mathbf{X} + \lambda I\right)^{-1} \left(\mathbf{X}^\top Y + \lambda\mu\right), \tag{2.80}$$

$$V_\theta = \sigma^2 \left(\mathbf{X}^\top \mathbf{X} + \lambda I\right)^{-1}. \tag{2.81}$$

Equations (2.80) and (2.81) is found by repeating the construction from Equation (2.69), but now with the parameters observed to be equal to the

prior mean μ with uncertainty equal to the prior variance (Gelman, Carlin, et al., 2013). The observations are again stacked in $\mathbf{y}$ and $\mathbf{X}$ as in Equation (2.55). If our prior expectation is equal to zero, then Equation (2.80) is identical to the ridge regression from Equation (2.68). We see that as λ is increased from zero, the solution changes from the ordinary least squares solution, in Equation (2.58), to become closer and closer to the prior.

The form of the posterior mean is recognized from the discussion on ridge regression and maximum likelihood in Sections 2.1.8 and 2.1.9. Recall the weighted least squares interpretation of ridge regression from Section 2.1.9, which was equivalent to having "observed" the parameters being equal to zero.

When the posterior distribution is not available in closed form, we must approach the problem differently. We consider three different strategies, point estimates of the posterior, fitting analytical distributions to the posterior, and sampling from the posterior.

We first consider point estimates of the posterior. In particular, the point estimate where the posterior takes its maximum value. We refer to this as the *maximum a posteriori* (MAP) estimate. This is similar to the maximum likelihood method studied in Section 2.1.8. Formulated as the minimization problem in Equation (2.64), we get the loss function

$$L(\theta) = -p(\theta|\mathcal{D}). \tag{2.82}$$

Similarly to maximum likelihood, it is convenient to consider the unnormalized log posterior, which yields an equivalent loss function

$$L(\theta) = -\log p(\mathcal{D}|\theta) - \log p(\theta). \tag{2.83}$$

We recognize the first term as the log-likelihood, and the second term is the log prior. We can work with the unnormalized posterior because the normalization term, $p(\mathcal{D})$, does not include the parameters θ and does not influence the optimization.

Recall how we in Section 2.1.8 derived the connection between maximum likelihood and least squares estimates, by making some assumptions about the model. We now repeat this exercise for the MAP estimate. Again consider the linear regression model in Equations (2.52) to (2.54). We consider the prior $\theta \sim \mathcal{N}(0, \sigma^2/\lambda I)$, with $\lambda > 0$. We can follow the same derivations as in Section 2.1.8, which again results in the log-likelihood from equation Equation (2.50). Similarly, we find

$$\log p(\theta) = p \log \sqrt{\frac{\lambda}{2\pi\sigma^2}} - \frac{\lambda}{2\sigma^2}\theta^\top\theta. \tag{2.84}$$

for the log-prior. Inserting Equation (2.50) and Equation (2.84) into Equa-

tion (2.83) we get

$$\log p(\theta|\mathcal{D}) = -n\log\sqrt{2\pi} - n\log\sigma - \frac{1}{2\sigma^2}\sum_{i=n}^{n}\left(y_i - f(x_i;\theta)\right)^2 + p\log\sqrt{\frac{\lambda}{2\pi\sigma^2}} - \frac{\lambda}{2\sigma^2}\theta^\top\theta, \quad (2.85)$$

which, when we focus on θ, yields the equivalent loss function

$$L(\theta) = \sum_{i=n}^{n}\left(y_i - f(x_i;\theta)\right)^2 + \lambda\theta^\top\theta, \quad (2.86)$$

when all the factors and terms independent of θ are removed. We recognize this as the regularized loss function from Equation (2.65), with λ from the prior appearing as the tuning factor. This gives us the connection between the normal prior that leads to Equation (2.85) and the squared norm regularization from Equation (2.66), and more generally the connection between the MAP estimate from Equation (2.83) and the regularized maximum likelihood estimate.

If we need to capture more of the posterior distribution than what is provided by the point estimate, we can attempt to fit an approximate distribution to match it. The simplest approach is to take a normal approximation at the posterior mode, by taking the MAP solution as the expected value and the negative inverse second derivative at this point as the variance (Gelman, Carlin, et al., 2013). An alternative approximation is *variational inference*, which is frequently used in machine learning. Variational inference constructs a parametric approximation of the posterior, $q(\theta)$ (Gelman, Carlin, et al., 2013). The parametric approximation belongs to a class of candidate densities $\mathcal{Q}$, and we seek the best approximation,

$$\arg\min_{q\in\mathcal{Q}} L(q), \quad (2.87)$$

according to some loss function $L : \mathcal{Q} \to \mathbf{R}$. The loss is often taken to be the Kullback-Leibler divergence (Blei, Kucukelbir, and McAuliffe, 2017),

$$L(q) = KL(q||p) = \mathbf{E}_q\left[\log q(\theta)\right] - \mathbf{E}_q\left[\log p(\theta|\mathcal{D})\right]. \quad (2.88)$$

Note that the Kullback-Leibler divergence is not symmetric, and the expectation is with respect to q. An important factor of this optimization problem is the class of densities $\mathcal{Q}$. The mean-field variational family is a common choice when the dimension of θ is large. The mean-field family approximates the density as the product of marginal distributions, $q(\theta) = \prod_{k=1}^{K} q_k(\theta_k)$. The marginals themselves can be any suitable distribution, with the Normal distribution being a common choice for continuous variables. Variational inference can be scaled to problems with a large number of parameters, for instance, deep neural networks. Such neural networks are often referred to as Bayesian neural networks.

Point estimates and fitted distributions are both optimization problems that attempt to capture the shape of the distribution directly. Simulation methods

take a different approach. Simulation methods exploit the fact that it is often feasible to generate samples from a distribution that otherwise is challenging to study analytically. The intuition behind simulation methods is that the histogram of a set of samples from a density can for many considerations be a good representation of the density itself, and the samples can be used to estimate quantities of interest such as moments or percentiles (Gelman, Carlin, et al., 2013). These quantities can be represented as the expected value of a function of the random variable in question. Let $\theta^{(1)}, \ldots, \theta^{(s)}$ be s samples simulated from the posterior distribution $p(\theta|\mathcal{D})$, We can then find the Monte Carlo estimate of the expected value of θ as the sample mean

$$\mu_{MC} = \frac{1}{s}\sum_{i=1}^{s}\theta^{(s)}, \tag{2.89}$$

which converges to the true expected value, $\mu = \int \theta p(\theta|\mathcal{D})d\theta$, as $s \to \infty$ (Givens and Hoeting, 2012). Similarly, quantities such as variance and percentiles are easily computed from the set of samples. In fact, we can change $\theta^{(s)}$ with any transform $h(\theta^{(s)})$ we might be interested in to approximate its expected value $\mathrm{E}[h(\theta)]$. There is a vast range of methods to generate samples, and the choice of methods is highly dependent on the structure of the model (Givens and Hoeting, 2012). Section 2.1.12 gives a brief review of the methods relevant to our application.

2.1.11 Numerical optimization

So far the discussion has been focused on problems we want to solve, without concern for *how* they can be solved. Many of the problems we have discussed ultimately take the form of an optimization problem. Examples are maximum likelihood and least squares estimates discussed in Section 2.1.8, and the MAP and distributional approximations to the posterior distribution discussed in Section 2.1.10. In section Section 2.1.8 we saw an example of least squares regression where we were able to derive an analytical expression for the minimum, but this is an exception rather than the norm. For most problems we are interested in, we must approximate the solutions to these optimization problems numerically. This is known as *numerical optimization*. We now review the most relevant numerical optimization techniques The discussion is based on Nocedal and Wright (2006) and Givens and Hoeting (2012).

Our problems originate from learning problems for statistical models. We formulate our general optimization problem using the notation established in Section 2.1.2 and Section 2.1.9, which differs slightly from the notation usually found in the numerical optimization literature. As such, we wish to optimize a quantity $\theta \in \Theta$, according to a loss function $L : \Theta \to \mathbf{R}$. In numerical optimization, we refer to the loss, L, as the *objective function*, which in our case describes the properties we wish our model to have, for instance, the negated log-likelihood Equation (2.48). Further, the parameter space Θ is referred to as the *search space*, and the elements of Θ as *candidate solutions*. Any restrictions

on the search space are referred to as *constraints*. Without loss of generality, we only consider minimization problems, which we write as

$$\arg\min_{\theta \in \Theta} L(\theta). \tag{2.90}$$

This is a very general problem statement, and significant complexity can be hidden in the objective function and the formulation of the search space.

Let θ be a vector of p elements, $\theta = \begin{bmatrix}\theta_1 & \dots & \theta_p\end{bmatrix}$. An optimization problem is called *continuous* if θ is real-valued, $\theta \in \mathbf{R}^p$, *discrete* if θ is integer-valued, $\theta \in \mathbf{N}^p$, and a *mixed integer* problem if it is a combination of integer and real-valued elements. An optimization problem is *unconstrained* if there are no limitations on the search space, for instance if $\Theta = \mathbf{R}^p$, and *constrained* if we have restrictions on the values θ can take. For instance, if $\theta \in \mathbf{R}^p$, but with the additional constraint $\sum_{k=1}^{p} \theta_k = 0$, it would be a equality constrained problem, or if we require that $\theta_k \geq 0,\ k = 1, \dots, p$, it would be an inequality constrained problem.

We now consider continuous, unconstrained optimization problems, with $\theta \in \mathbf{R}^p$. How hard the problem in Equation (2.90) is to solve depends on the structure of $L(\theta)$. There may be one or more minima. A point $\theta^\star$ is a *global minimizer* of Equation (2.90) if $L(\theta^\star) \leq L(\theta)$ for all $\theta \in \Theta$, and a *local minimizer* if the same condition holds for all θ in a neighbourhood around $\theta^\star$. For some problems, we can prove that there is at most one local minimum, which then coincides with the global minimum. This is, for instance, the case if L is *convex* (Boyd and Vandenberghe, 2004), which means that for any two points $\theta^{(a)}, \theta^{(b)} \in \Theta$, we have

$$L(\alpha\theta^{(a)} + (1-\alpha)\theta^{(b)}) \leq \alpha L(\theta^{(a)}) + (1-\alpha)L(\theta^{(b)}), \tag{2.91}$$

for all $\alpha \in [0, 1]$. To effectively recognize a local minima we assume that the objective function is twice continuously differentiable. Let $\theta^\star \in \Theta$ be a candidate point. The first necessary condition for $\theta^\star$ to be a minimum is that it is a critical point,

$$\frac{\partial}{\partial \theta} L(\theta^\star) = 0. \tag{2.92}$$

The secondary necessary condition is that its second derivative is positive semi-definite,

$$\frac{\partial^2}{\partial \theta^2} L(\theta^\star) \geq 0. \tag{2.93}$$

We get a sufficient condition if Equation (2.93) is changed to a strict inequality, meaning that the second derivative is positive definite.

As an example, Let $\theta \in \mathbf{R}$ and $L(\theta) = \mathbf{x}^2$. We have one critical point at $\theta = 0$, because $\frac{\partial}{\partial \theta} L(\theta) = 2\theta$ is zero for $\theta = 0$. This point also satisfies the secondary sufficient condition, because $\frac{\partial^2}{\partial \theta^2} L(\theta) = 2 > 0$. We therefore can conclude that

$\theta = 0$ is a local minimum of this function. If we instead consider $L(\theta) = \mathbf{x}^3$, we still have a critical point at $\theta = 0$, but it is not a local minimum. It satisfies the secondary necessary condition, but not the sufficient condition. Further, consider $L(\theta) = \mathbf{x}^4$. This also has a critical point at $\theta = 0$ that satisfies the secondary necessary condition, but not the sufficient condition. However, this time it is a local minimum. These examples illustrate how the necessary conditions can be too relaxed and the sufficient condition can be too strict. When L is convex it is sufficient that $\theta^\star$ is a critical point. This was the case for the least squares estimate with a linear regression model from Equation (2.58), which is why it was sufficient to find a point that satisfied Equation (2.92). In the examples above, x^2 and x^4 are convex, while x^3 is not. This is one of the reasons why convexity is such an important concept in optimization.

For problems without an analytical solution, the solution can be *approximated* by numerical optimization algorithms. These algorithms are iterative procedures, that incrementally improve a best guess $\theta^{(k)} \in \Theta$ over a sequence of steps $k = 1, 2, \ldots$, until sufficient accuracy or a maximum number of steps is reached. Just as the diversity in optimization problems is great, we have great variety in the available algorithms. The choice of optimization algorithm depends on the problem at hand. Generally, a closer coupling between the problem and the algorithm will lead to solutions with better properties and faster run times (Givens and Hoeting, 2012). The algorithms differ in the type of information they utilize during the iterations. In addition to the objective function, $L(\theta)$, some algorithms also utilize the *gradient*,

$$G(\theta) = \frac{\partial L}{\partial \theta}(\theta), \tag{2.94}$$

which takes values in $\mathbf{R}^p$, and the *Hessian*,

$$H(\theta) = \frac{\partial^2 L}{\partial \theta^2}(\theta), \tag{2.95}$$

which takes values in $\mathbf{R}^{p \times p}$. Evaluating these quantities can be computationally expensive, and the choice of including this information is a tradeoff between the quality of each iteration and the speed at which iterations can be performed.

An important class of optimization algorithms is based on the update equation

$$\theta^{(k+1)} = \theta^{(k)} + \alpha^{(k)} v^{(k)}. \tag{2.96}$$

The update consists of a search direction $v^{(k)} \in \mathbf{R}^p$ and a step length $\alpha^{(k)} \in \mathbf{R}$, with $\alpha^{(k)} > 0$. In machine learning, the step length is sometimes referred to as the *learning rate*. It is common to take $v^{(k)}$ to be a *descent direction*, which means that its inner product with the gradient of the objective function is negative, $G(\theta^{(k)})^\top v^{(k)} < 0$. *Gradient decent* takes the descent direction to be the negative of the gradient itself, which trivially satisfies this condition, $-G(\theta^{(k)})^\top G(\theta^{(k)}) < 0$. Second order methods also incorporate second-order derivatives, such as *Newton's method*, which find the search direction as the solution of $H(\theta^{(k)})v^{(k)} = -G(\theta^{(k)})$, which is $v^{(k)} = -H(\theta^{(k)})^{-1}G(\theta^{(k)})$ if

the inverse Hessian exists. Augmentations to the Hessians exist to ensure positive definiteness. Incorporating exact second-order information can be computationally infeasible for large problems.

After finding a search direction, the algorithm needs to determine how far it should go before a new direction is selected at the next iteration. This is just as essential as the choice of the direction. We illustrate the importance with a simple example. Consider again the case $\theta \in \mathbf{R}$ with objective function $L(\theta) = \theta^2$, with an arbitrary starting point $\theta^{(0)} \in \mathbf{R}$. Gradient descent gives us the updated equation

$$\theta^{(k+1)} = \theta^{(k)} - \alpha^{(k)} G(\theta^{(k)}) \tag{2.97}$$

$$= \theta^{(k)} - 2\alpha^{(k)}\theta^{(k)} \tag{2.98}$$

$$= (1 - 2\alpha^{(k)})\theta^{(k)}. \tag{2.99}$$

If we choose a fixed value for the step length, $\alpha_k = a$, we can get three different dynamics. For $a = 1$ the iterates will alternate between $\theta^{(0)}$ and $-\theta^{(0)}$, for instance, $\theta^{(0)} = 1$, $\theta^{(1)} = -1$, $\theta^{(2)} = 1$, and so on, without any progress. For $a > 1$ the iterates will *diverge*, meaning that the iterations will take on larger and larger values, for instance with $a = 2$ we get $\theta^{(0)} = 1$, $\theta^{(1)} = -2$, $\theta^{(2)} = 4, \ldots$, $\theta^{(k)} = (-2)^k$. And lastly, $a < 1$ the iterates will *converge* towards zero, at the rate $\theta^{(k)} = (1 - 2a)^k\theta^{(0)}$. The choice $a = 1/2$ is particularly interesting, as this makes the iterations converge in one step, regardless of starting the initial value, $\theta^{(0)} = 1$, $\theta^{(1)} = 0$, $\theta^{(2)} = 0$, and so on. Compare this with the Newton's method, where the update equation becomes $\theta^{(k+1)} = (1-\alpha^{(k)})\theta^{(k)}$, for which the optimal step length is $\alpha^{(k)} = 1$. Consider now a different objective function, $L(\theta) = \theta^4$. The gradient descent iterations now become $\theta^{(k+1)} = \theta^{(k)} - \alpha^{(k)} 4(\theta^{(k)})^3$, for which the ideal choice of step length depends on the value of $\theta^{(k)}$.

The examples above illustrate the importance of selecting the appropriate step length, and that this choice is sensitive to both the choice of search direction and the objective function. Given a search direction, the optimal step length would be

$$\arg\min_{\alpha^{(k)} \in \mathbf{R}} L(\theta^{(k)} + \alpha^{(k)} v^{(k)}), \tag{2.100}$$

but solving this problem would generally be too expensive to do each iteration. Instead, strategies such as *back-tracking line search* start with an initial guess of the step length and iteratively reduce it in size until a sufficient decrease in the objective is found.

However, for many large-scale machine learning problems, the sheer number of data points and parameters makes a careful selection of search direction and step length computationally impractical. *Stochastic gradient descent* is a class of methods that attempts to overcome this Bottou, Curtis, and Nocedal (2018). They use an *estimate* of the gradient, based on a sub-sample of data, to reduce the computations involved in selecting the search direction. This means that $v^{(k)}$ is a random variable with $\mathbf{E}[v^{(k)}] = G(\theta^{(k)})$. Additionally, stochastic gradient

descent usually employs a predefined step length sequence, $\alpha^{(k)}, k = 1, 2, \ldots$, to further reduce the number of function evaluations. If the sequence is constructed such that

$$\sum_{k=1}^{\infty} \alpha^{(k)} = \infty, \tag{2.101}$$

$$\sum_{k=1}^{\infty} (\alpha^{(k)})^2 < \infty, \tag{2.102}$$

then the stochastic gradient descent converges to a local minimum with probability one (Blum, 1954). Robbins and Monro (1951) suggested the sequence $\alpha^{(k)} = a/k,\ a > 0$ as a sequence that satisfies Equations (2.101) and (2.102). It is a dramatic contrast between the examples that converged in one step and the potentially unbounded sequence of stochastic gradient descent. However, empirically, these methods have proved to be quite efficient and reliable, and they have been particularly useful in the context of training neural networks (Goodfellow, Bengio, and Courville, 2016). A part of the success is due to the computational speed of these stochastic gradient methods, which makes it feasible to perform many iterations on limited hardware, Another reason may be their ability to escape local critical points due to the randomness is the search direction. The benefit of such randomness in the updates is well established for other optimization algorithms such as simulated annealing (Givens and Hoeting, 2012).

The methods above make use of the gradient of the objective function. We have seen examples where these gradients are trivial to provide, but this is not always the case. For instance, for neural networks, we could provide these gradients through *back-propagation*, which is the repeated application of the chain rule of calculus (Goodfellow, Bengio, and Courville, 2016). This makes the gradients for a layer the product of partial derivatives from layers above it. For deep neural networks, this can be problematic for the first layers, because a single partial derivative can make the gradients very close to zero or arbitrarily large. This is known as vanishing and exploding gradients (Nalisnick, Smyth, and Tran, 2023). Several measures can be taken to alleviate this problem, for instance, the use of suitable activation functions, such as the rectified linear unit, or skip connections, as discussed in Section 2.1.5. However, to effectively compute gradients in practice, we would implement these models in software libraries that support automatic differentiation of computational graphs (Paszke et al., 2019), which significantly simplifies the effort needed to apply gradient descent methods.

Derivative-free algorithms is an alternative to the gradient-based methods discussed above (Conn, Scheinberg, and Vicente, 2009). In derivative-free optimization the next iteration is decided based on the sequence of points seen so far, meaning that we find $\theta^{(k)} = g_k\left((\theta^{(k-1)}, L(\theta^{(k-1)}), \ldots, (\theta^{(1)}, L(\theta^{(1)})\right)$, for an appropriate sequence of functions, $g_k, k = 1, 2, \ldots$. One approach that is common in hyper-parameter optimization is *random search*, where candidate

points are randomly drawn from the search space (Bergstra and Bengio, 2012). Random search makes no assumptions about the objective function, which makes it easy to use. The downside is the lack of any guarantees on the result. Malherbe and Vayatis (2017) presents a derivative-free method capable of approximating a *global solution*, by assuming that the objective function has a finite Lipschitz constant. This means that for any $\theta_1, \theta_2 \in \Theta$, the function satisfies $||f(\theta_1) - f(\theta_2)|| \leq c||\theta_1 - \theta_2||$ for a Lipschitz constant $c \geq 0$. This allows the algorithm to maintain a lower bound on the solution over the entire search space Θ, and compare this to the currently best seen candidate point. The iterations are based on selecting the next point as one that has a lower bound on the objective function than the current best point. Derivative-free methods will generally converge slower than gradient-based methods (Conn, Scheinberg, and Vicente, 2009).

Other global optimization methods also require assumptions on the objective function to guarantee that a global optimum has been found with sufficient accuracy. For instance, the method of Grimstad and Sandnes (2016) assumes that the functions involved are represented by splines, and use the properties of splines to generate increasingly tight lower bounds of the solution until it can be confirmed that the best local minimum seen is also the best global minimum.

The algorithms discussed above have focused on continuous optimization problems. However, it is often the case that we need the solution to take on discrete, integer values. This is for instance the case with many of the hyper-parameters in the models we discussed in Section 2.1.7, such as the number of basis terms in Equation (2.11) or the width and depth of a neural network. Methods based solely on a decent strategy will not be able to satisfy integer constraints directly. However, other methods may be augmented to do so. For instance, the random search of Bengio (2012) is trivially augmented to discrete values, as the random candidates can be drawn directly from a discrete distribution. The method of Malherbe and Vayatis (2017) can also in practice be augmented to discrete candidates (King, 2009).

2.1.12 Monte Carlo methods

In Section 2.1.10 we discussed the posterior distribution of $\theta \in \Theta$ given a dataset $\mathcal{D}$, $p(\theta|\mathcal{D})$, and presented various methods for studying this distribution. Among these were *Monte Carlo* methods. Monte Carlo methods approximate the quantities of interest by the use of random samples simulated from the distribution in question. While we are interested in the posterior distribution, the Monte Carlo method can be used to study any distribution of interest.

The essence of the Monte Carlo approach is that samples, $\theta^{(1)}, \ldots, \theta^{(s)}$, simulated from the posterior can be used in inference such as the expectation in Equation (2.89). This section gives a brief introduction to sampling methods, with emphasis on methods for state-space models.

Let $p(\theta|\mathcal{D})$, as defined above, be the *target distribution* that we want to sample from. Simulation methods can be both exact and approximate (Givens and Hoeting, 2012). Exact methods simulate samples from the true target

distribution. This is for instance the case for all familiar standard distributions, such as uniform and Normal distributed values, which are easily generated by most statistical software libraries. However, most of the time we are not able to sample directly from $p(\theta|\mathcal{D})$. We may, instead, be able to evaluate the density, or an unnormalized version of the density, for given values of θ. This is often exploited in conjunction with a *proposal distribution*, $q(\theta)$, which is easier to sample from. For instance, *Rejection sampling* simulates from the proposal distribution and then accepts or rejects the samples randomly, such that the relative frequency of the accepted samples is equal to that of the target distribution. Rejection sampling is exact but can be slow if the rejection rate is high.

Sampling from exact methods is of course ideal, but it can be computationally infeasible for non-trivial distributions. For some cases, it is therefore beneficial to generate many samples from an inexact method, instead of a few samples from an exact method. Similar to rejection sampling, approximate methods simulate their samples from a proposal distribution, $q(\theta)$, and proceed to adjust or discard the generated samples to make them resemble the target distribution. However, in contrast to rejection sampling, this adjustment is not exact.

Sampling importance resampling (SIR) is an approximate sampling method (Givens and Hoeting, 2012). It first generates m candidate samples, $\theta^{(1)}, \ldots, \theta^{(m)}$, from a proposal distribution $q(\theta)$, and the computed weights corresponding to each sample. For the SIR algorithm, the weights are computed as

$$w^{(k)} = \frac{p(\theta^{(k)}|\mathcal{D})/q(\theta^{(k)})}{\sum_{i=1}^{s} p(\theta^{(i)}|\mathcal{D})/q(\theta^{(i)})}, \tag{2.103}$$

for $k = 1, \ldots, m$, which is referred to as standardized *importance weights*. The approximate sample is then obtained by resampling s samples, with replacement, from the candidates, with probabilities equal to the importance weights. The distribution of the final s samples converges to the target distribution as $m \to \infty$. It is generally beneficial to have $s < m$ (Givens and Hoeting, 2012).

Sequential Monte Carlo is an adaptation of SIR that is especially suited for state-space models. Recall from our discussion of state-space models from Section 2.1.7 that these models are recursively defined, and we wish to infer a sequence of hidden state variables, $z_t \in \mathcal{Z}$, for $t = 0, 1, \ldots, n$, given a sequence of observed data $\mathcal{D}_t$ for $t = 1, \ldots, n$.

We use the notation $z_{a:b} = \{z_a, z_{a+1}, \ldots, z_{b-1}, z_b\}$ to indicate subsets of the states, and similarly $\mathcal{D}_{a:b}$ for the observations. The model consists of the initial state $p(z_0)$, a transition equation $p(z_{t+1}|z_t)$, an a likelihood $p(\mathcal{D}_t|z_t)$ (Doucet, De Freitas, and Gordon, 2001). The goal is to obtain samples from the posterior distribution $p(z_{0:t}|\mathcal{D}_{1:t})$. The posterior distribution can be factorized recursively as,

$$p(z_{0:t}|\mathcal{D}_{1:t}) = p(z_{0:t-1}|\mathcal{D}_{1:t-1}) \frac{p(\mathcal{D}_t|z_t)p(z_t|z_{t-1})}{p(\mathcal{D}_t|\mathcal{D}_{1:t-1})}, \tag{2.104}$$

which is exploited by the sequential Monte Carlo algorithm to efficiently process the data from each time step successively. Samples are simulated from a proposal

distribution, $q(z_t|z_{t-1})$, and assigned the unnormalized importance weights

$$w_t = \tilde{w}_t w_{t-1}, \tag{2.105}$$

where the increments are given by

$$\tilde{w}_t \propto \frac{p(\mathcal{D}_t|z_t)p(z_t|z_{t-1})}{q(z_t|z_{t-1})}. \tag{2.106}$$

A set of s samples, $z_{0:t}^{(1)}, \ldots, z_{0:t}^{(s)}$, can be generated one by one or in parallel (Givens and Hoeting, 2012). These can then be resampled similarly to the SIR algorithm. It is common to generate all samples in parallel, processing one observation at a time. Resampling can be done at each time step or when necessary. Note that in the SIR algorithm, the resampling is done after all the samples are generated, while in sequential Monte Carlo the resampling occurs during the generation at various time steps. The purpose of this resampling is to ensure that the samples do not degenerate, which happens when the weights place all the importance on a small subset of the samples. This typically happens when the sequence of steps is long. After resampling, the unnormalized importance weights are reset to one, and the sampling procedure is continuous. While resampling alleviates the degeneracy of the samples at the current time step, it will cause deterioration of the past state estimates, by reducing the variability of these quantities. Care must therefore be taken when designing an sequential Monte Carlo procedure to ensure that the resulting estimates are suitable for the problem at hand.

The main challenge with sequential Monte Carlo, apart from formulating the state space model itself, is to construct a well-behaved proposal distribution. Desirable properties are that it is easy to sample from and that the resulting sample weights have a low variance. This is achieved by making the proposal distribution as close to the product of the likelihood and transition distribution as possible. An often trivial proposal distribution is the transition distribution, $q(z_t|z_{t-1}) = p(z_t|z_{t-1})$. This is known as the Bootstrap filter (Doucet, De Freitas, and Gordon, 2001). This makes the weights equal to the likelihood, $p(\mathcal{D}_t|z_t)$, by canceling the factors in Equation (2.106). The Bootstrap filter can suffer from poor performance if the likelihood is very informative. The problem is that this leads to a large number of proposals with very low importance weights, which leads to a degenerate set of samples, as discussed above. In such cases, it is beneficial to construct proposals that also take part of the likelihood into consideration.

2.2 Soft sensing

Soft sensing is *real-time utilization of mathematical models to augment existing physical sensors in an industrial process* (Fortuna et al., 2007). The exact definitions vary in the literature (Jiang et al., 2021), but the essence is the application of software-based methods to compute quantities of interest in an

industrial process. There has been an increasing interest in soft sensing, which has partially been fueled by the increasing amount of hardware-based sensor data that has been made available in real-time. Additionally, their popularity is motivated by lower cost, ease of installation, and the possibility to estimate hard-to-measure quantities (Jiang et al., 2021).

Applications of soft sensing include backup of physical measuring devices, reducing the need for hardware installed in potentially hostile environments, and estimation for monitoring and control (Fortuna et al., 2007). Additionally, the models derived for soft sensor purposes can be utilized in what-if studies (Fortuna et al., 2007), a synergy that further improves their economic viability when compared to installing additional hardware-based measuring devices.

Soft sensors are based on either mechanistic or data-driven models, or a combination of the two (Fortuna et al., 2007; Jiang et al., 2021). Data-driven methods have seen a surge in popularity recently, due to the fact that many industrial processes are too complex to accurately model from first principles (Fortuna et al., 2007). The methods applied in data-driven soft sensing models cover a broad range of statistical inference and machine learning techniques (Sun and Ge, 2021). Harrou et al. (2021) compares ten different statistical and machine learning models, ranging from linear least squares regression to dynamic neural networks on an industrial application. The models achieve comparable performance, and they conclude that for a case with moderate quantities of data, the simpler model types are at an advantage when compared to deep learning methods. However, deep learning methods have seen an increase in popularity for soft sensing, due to their synergy with increasing data quantities and the complex nonlinearities involved in the processes (Sun and Ge, 2021).

The industrial process being modeled is, generally, a dynamic system, with components that may be placed far away from each other. It is therefore not uncommon to have significant delays between the time a change is measured at one location and the effect in the response is detected at another location, either due to transport delays, reaction times, or other phenomena. This can complicate the design of soft-sensing models (Qiao, Zhou, and Meng, 2023).

Consider a process that is observed over a sequence of time steps, $t = 1, 2, \ldots$, for which we always have a corresponding set of measurements $x_t \in \mathcal{X}$ available. Let $y_t \in \mathcal{Y}$ be the quantity of interest, and that this quantity is *not* always available. For simplicity, assume that we historically have observed pairs of data $\mathcal{D} = \{(x_t, y_t)\}_{t=1}^{n}$, and that for a future time step $t = T$ we only have access to x_T. We wish to study y_T through the posterior predictive distribution

$$p(y_T|x_T, \mathcal{D}). \tag{2.107}$$

Consider first the case where the measurements available at time t, x_t, conveys enough information about the process' current state to predict y_t. This allows us to consider *static models*, $p(y_T|x_T, \theta)$ (Jiang et al., 2021). In this case, we may use the historical data to infer a set of model parameters, and then only consider the parameters, and not the data, when performing future predictions. This enables the use of the statistical models discussed in Section 2.1.4. An

example of this is the feedforward neural network presented by (Gonzaga et al., 2009) for monitoring and control of an industrial process. Qiao, Zhou, and Meng (2023) presents a static model where the explanatory variables are *time delayed* versions of the actual measurements. The time delays were found individually for each measurement. This allows the static model to predict a response with a fixed time delay from the measurement location of the explanatory variables.

While static models can be convenient, they have natural limitations, for instance, if the response is related to the rate of change of one of the measured quantities. A trivial example is that we wish to predict the speed of an object, but only have access to a positional measurement, in which case we would need a sequence of measurements to succeed. A solution is to consider a model that takes a window of past observations, $p(y_T|x_T, x_{T-1}, \ldots, x_{T-k},)$, as its explanatory variables. The moving window convolutional neural network model of Shi et al. (2021) is an example of this approach. It was motivated by a time delay between the explanatory variables being measured and the effect being observable in the response.

If the dynamic aspects of the process are significant and a fixed sequence of measurements fails to provide sufficient information about the system state, a *dynamic model*, as discussed in Section 2.1.7, would be necessary. For instance, recurrent neural networks have been considered for several soft-sensing applications (Harrou et al., 2021; Huang et al., 2023).

While the amount of data collected from industrial processes is increasing, data-driven methods for soft sensing have two fundamental problems. The first is that *labeled* data is still scarce, which means that we actually have a response y_t to pair with the explanatory variables x_t. This is because acquiring labeled data may be time-consuming, involve expensive experiments, or disrupt stable operations (Curreri, Patanè, and Xibilia, 2021). The second is that working conditions change over time, which means that the data distributions are subject to change (Curreri, Patanè, and Xibilia, 2021), and there is limited *relevant* information in the available historical data to describe current and future operating conditions (Jiang et al., 2021). Consider the prediction of a new data point as in Equation (2.107), which is based on past observations. The problem of relevant information means the predictive performance of the model may improve if we base the inference on a subset of the full dataset, because older, and outdated, observations may guide our inference toward a behaviour that is no longer present in the system. This may make it difficult to acquire enough data to train a satisfactory model in practice.

Transfer learning, multitask learning, and similar strategies are presented as a way to address these data issues, as it may allow us to incorporate more relevant data into the model training, but this has only been an active research area for a few years (Curreri, Patanè, and Xibilia, 2021). We consider two types of knowledge transfer between soft sensors. Between similar systems and between different operating conditions and states from the same system. Maschler and Weyrich (2021) refers to these as *cross-entity* and *cross-state* respectively. The strategies found in the soft sensing literature approach multitask learning from different angles, and no standards have emerged yet.

Recall classic multitask neural network of Caruana (1997), given in Equation (2.33), where we have m tasks, $j = 1, \ldots, m$. This architecture learns a shared set of basis function $h : \mathcal{X} \to \mathbf{R}^p$, and each task performs a regression on this basis to find task parameters $\beta \in \mathbf{R}^p$. This lead to the predictions $f_j(x_{tj}) = h(x_{tj})^\top \beta_j$, given datapoint t from task j. Several multitask learning strategies for soft sensing are variations on this architecture. For instance, S. Yan and X. Yan (2020) presents a variation with soft parameter sharing, where each task is given its own neural network, $h_j : \mathcal{X} \to \mathbf{R}^p$, with predictions $f_j(x_{tj}) = h_j(x_{tj})^\top \beta_j$, and the output of the individual task networks, h_j, are forced to be correlated through a parameter regularization mechanism where the degree of regularization bridges the gap from completely independent to fully correlated basis expansions. Qiao, Zhou, and Meng (2023) suggests a strategy where measurements from each task are used as input to separate networks, $h_j : \mathcal{X} \to \mathbf{R}^p$, and the output of these individual networks are concatenated together to provide a shared set of basis functions for all tasks. Similar to the shared basis approach, the tasks have additional parameters $\beta_j \in \mathbf{R}^{mp}$, which enters as $f_j(x_{t,1}, \ldots, x_{t,m}) = \begin{bmatrix} h_1(x_{t,1})^\top & \ldots & h_m(x_{t,m})^\top \end{bmatrix} \beta_j$. This architecture assumes a tight coupling between the tasks, as they must all provide measurements in order to evaluate the model. In the example of Qiao, Zhou, and Meng (2023), the tasks are time lagged versions of *the same* measurements.

Both of the strategies above have an individual neural network for each task, and the multitask aspect is how the outputs of these networks are tied together. This enables us to consider tasks with different domains, $\mathcal{X}_j$, with neural networks $h_j : \mathcal{X}_j \to \mathbf{R}^p$ for $j = 1, \ldots, m$. When the networks are not forced to consider the same inputs we can model tasks with different levels of instrumentation.

Finding the optimal level of knowledge transfer is challenging. The closer the tasks are tied together, the higher the risk of negative transfer and performance deterioration. Huang et al. (2023) presents a solution to negative transfer where each task is allowed to learn how much it is influenced by different sources of information, by the use of a multitask extension to the mixture of experts model. Here, each task is given its own gating function, which creates a unique expert weighting for each task. This combination of experts is then used as inputs to a task-specific neural network. Essentially, this allows each task to have multiple task-specific layers, as opposed to the single layer in Equation (2.33). Pan et al. (2023) further extends this methodology by also allowing dynamic weighting of the individual tasks in the loss function during training.

Some of the architectures above can grow large in terms of parameters, especially when there are many tasks involved. This can be an issue if the soft sensors must run on limited hardware in a distributed, edge computing setting. Zhai et al. (2023) presents a strategy to cope with this based on knowledge distillation, where they first train a large shared multitask neural network, and then allow individual tasks to train smaller networks that are regularized towards the predictions of the shared network.

Regardless of the modeling strategy, the models require maintenance to avoid

performance decline. Since there is no wear and tear on soft sensors, the driving forces behind model maintenance are changes to the system that is being modeled or changes to its operating points. As discussed above, unless the historical data used during model creation contained information about all possible future states of the system, it is unlikely that the model will retain its performance when changes occur, regardless of how well it was constructed (Kadlec, Grbić, and Gabrys, 2011). Updating existing models as new data arrives is often referred to as *(re-)calibrating* or *(re-)tuning* the models (Fortuna et al., 2007; Jiang et al., 2021). This typically involves a subset of the steps taken during the initial model creation (Jiang et al., 2021).

2.3 Flow rate measurements in oil and gas production networks

Our soft sensing application is the measurement of flow rates in oil and gas production networks. This section gives a brief introduction to such networks, the commonly used measuring techniques, and the value of such measurements.

An oil and gas asset can be viewed as a network of wells, pipelines, and a processing facility. Wells are drilled into a reservoir which contains a mixture of oil, gas, and water. This gas and liquid mixture is transported via a network of pipelines up to the surface and into a processing platform, where gas, oil, and water are physically separated into three individual streams. A simplified network is illustrated in Figure 2.2. The components of a fluid mixture are commonly referred to as *phases*. A fluid with only one component, for instance, pure gas, is called a single-phase flow. A mixture of two or more phases, for instance, oil and water, is called multiphase flow.

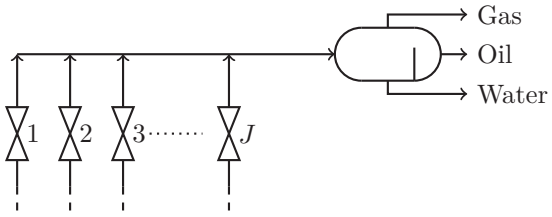


Figure 2.2: Asset with J wells connected to the same pipeline leading to a single separator. Here, the wells are represented by their choke valve, which is further illustrated in Figure 3.1. At the broken lines, the wells extend into a reservoir with a mixture of gas, oil, and water. The choke valve controls the flow from each well into the shared separator. This figure originally appeared in Paper II.

The end product of the asset described in Figure 2.2 is the separated single-phase flow of oil and gas. The economic incentive is to maximize these flow rates, both in terms of instantaneous and cumulative production. Knowledge of flow rates throughout the system is key to addressing these optimization

problems. On the cumulative side, such knowledge leads to better-calibrated reservoir models, which enables planning of development and recovery. On the instantaneous side, it enables the optimization of bottlenecks in the production system, such as limitations on water processing or gas lift allocations. Flow rates are also crucial in flow assurance operations, which are concerned with keeping the system operating in a safe and maintainable state. While the flow rates throughout the entire system can be of interest, the flow rates from individual wells are often the most sought after, because it gives the most complete picture of the production state. We therefore focus our attention on well flow rates.

2.3.1 Physical measuring devices

The most reliable way to measure flow rates is after separation. Single-phase measurements are the most precise, with errors as small as 0.25% for oil and 1% for gas (Thorn, Johansen, and Hjertaker, 2012). Water may be less accurately measured, and is highly asset dependent. The single-phase measurements taken after separation provide a continuous measure of the combined production from all wells routed to the separator. To measure an individual well this way, it can be routed to a second, smaller separator, where it is left alone. This solitary measurement is known as a *well-test* and is the gold standard for measuring the flow from a well. However, each test may take several hours, making well-tests a sparse source of information (Monteiro et al., 2020).

Limitations to well-testing create an incentive to measure flow rates *before* separation. Multiphase flow is significantly more challenging to measure than a single phase, due to its complex dynamics and uncertain fluid properties (Jansen, 2015). Sensors for multiphase flow are divided into two main classes, multiphase flow meters and virtual flow meters (Bikmukhametov and Jäschke, 2020).

Multiphase flow meters are physically installed on-site. They are complex devices that rely on several measurement principles to infer flow rates. A properly calibrated multiphase flow meter can be expected to achieve 5% error for gas, oil, and water rates (Thorn, Johansen, and Hjertaker, 2012), but they are subject to drift and will require re-calibration if operating conditions change (Corneliussen et al., 2005). Due to the need for a physical device to be installed in the well, multiphase flow meters can be quite expensive to acquire and maintain.

2.3.2 Virtual flow meters

Virtual flow meters are a soft sensor alternative to multiphase flow meters. virtual flow meters use mathematical models to infer flow rates from existing measurements (Toskey, 2012). Because of this, the models can vary significantly in terms of explanatory variables and responses. With enough information, they can attempt to infer all three flow phases, but often they are limited to a single phase or the sum of all phases (Bikmukhametov and Jäschke, 2020). In scenarios with limited information, it is possible to assume knowledge of the *flow composition*, for instance, how much of the flow is gas.

As with all soft sensing models, virtual flow meters can be based on *mechanistic* models, *data-driven* models, and *hybrid* models (Mathilde, Bjarne, and Imsland, 2020), where all model types can have dynamic or steady-state formulations. Solle et al., 2017 expands on the different model types.

The literature presents many candidates for data-driven models. This includes linear regression (Bello, Ade-Jacob, and Yuan, 2014), regression trees (Bikmukhametov and Jäschke, 2019), and neural networks (Shaban and Tavoularis, 2014). Variations on neural networks are the most common in the recent literature (Bikmukhametov and Jäschke, 2020). Several of the mentioned examples make use of dimensionality reduction techniques as part of model inference. The reason is that many of the available explanatory variables are co-linear. Common for all these data-driven strategies is that they build one model for each well, without sharing information between them.

The performance of virtual flow meters is naturally highly reliant on the quality and quantity of data available for calibration. In the literature, we frequently find errors in the range of 2–6% being reported for different data-driven methodologies (Bikmukhametov and Jäschke, 2019; AL-Qutami et al., 2017; AL-Qutami et al., 2018). However, it is optimistic to expect the performance seen on static datasets to carry over to real-time applications in practice, due to the nature of the data and process being modeled. Chapter 4 expands on this and explores why it is difficult to properly quantify the expected performance of these models. The uncertainty stems from both the model and the measurements (Hüllermeier and Waegeman, 2021). Proper estimation and reduction of these uncertainties has been highlighted as an important research direction for virtual flow meters (Bikmukhametov and Jäschke, 2020) and would be important for engineers in their decision-making.

Chapter 3

Virtual flow meter modeling strategy

This chapter describes the proposed strategy for a data-driven virtual flow meter model. First, Section 3.1 gives an introduction to the available measurements and the assumptions that are made. Then, Section 3.2 presents the architecture of the virtual flow meter models. Finally, Section 3.3 gives an outline of the proposed solution, and how the conducted research takes steps towards this.

3.1 Setup

We focus our attention on modeling flow through the well choke valve. The primary purpose of the choke valve is to control the flow rates from each well. Choke valves are chosen because they have a quite consistent set of instrumentation across the assets from which our data originates. The particular setup we consider is illustrated in Figure 3.1. The measurements are temperature, choke opening, and pressure on both sides of the choke. These are continuously measured and are our primary explanatory variables.

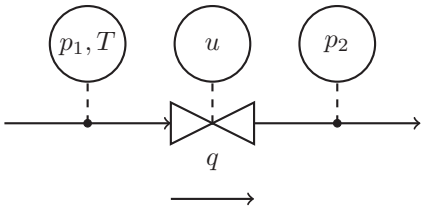


Figure 3.1: Topology of a well with instrumentation. The measurements are located at both sides of the choke valve and consist of upstream pressure p_1, downstream pressure p_2, temperature T, and choke opening u. The flow rate q is a mixture of gas, oil, and water, and is generally unknown. Figure originally appeared in Paper II.

The gas, oil, and water rates through the choke, denoted $q^\top = \begin{bmatrix} q_G, q_O, q_W \end{bmatrix}$, are unknown, except when measured by a well-test. Ideally, the virtual flow meter would be able to estimate all three rates at all times. However, it is generally not possible to infer all three based on a single pressure drop, such as a choke valve. Therefore, we focus on the sum of all rates,

$$y = q_G + q_O + q_W. \tag{3.1}$$

Additionally, it is assumed that the flow composition, $\phi = \frac{q}{y}$, is known to the virtual flow meter. Following the formulation of regression models in Equation (2.3), a virtual flow meter for a given well is then any function

$$y = f(x, \phi) + e, \tag{3.2}$$

where $y \in \mathbf{R}$ is the sum of the flow rates, $x \in \mathbf{R}^4$ is the measured choke position, pressures, and temperature, and $\phi \in [0, 1]^3$, is the flow composition. The error term, $e \in \mathbf{R}$, captures the process and measurement noise. The individual flow rates are found as $q = y\phi = \phi f(x, \phi)$. Equation (3.2) is a static model formulation, in the sense that we do not consider any dynamics of the system, and the model is simply a mapping from the inputs to the output (Solle et al., 2017).

The flow composition, ϕ, is primarily a function of the reservoir state, which is slowly changing over time. In many cases, the composition evolves slowly enough that it can be assumed constant between well-tests. However, if the frequency of testing is low, a dynamic model of its development is proposed.

Flow composition is typically represented by two quantities. It is convenient if these quantities are in the unit interval and do not depend on each other. Therefore, we choose to use the gas fraction,

$$\gamma = \frac{q_G}{q_G + q_O + q_W}, \tag{3.3}$$

$$\tag{3.4}$$

which is found as the gas rate divided by the sum of all three flow rates, and the oil factor,

$$\lambda = \frac{q_O}{q_O + q_W}, \tag{3.5}$$

which is the oil rate divided by the sum of oil and water rates. These are related to the composition vector by

$$\phi = \begin{bmatrix} \gamma \\ (1-\gamma)\lambda \\ (1-\gamma)(1-\lambda) \end{bmatrix}. \tag{3.6}$$

The flow composition quantities are undefined when the flow rates involved are zero. We define $\lambda = 0$ when $q_O = q_W = 0$, and similarly for γ. This has no practical consequences.

3.2 Model architecture

Our task is to identify the function f in Equation (3.2), using our prior knowledge about the process itself and the available data, and be able to provide it with the necessary flow composition, ϕ.

This research has been investigating the use of multitask learning to create virtual flow meter models. For this, we consider a set of m wells, indexed by $j = 1, \ldots, m$. Each well has well-test data

$$\mathcal{D}_j = \{(x_{ij}, q_{ij})\}_{i=1}^{n_j}, \tag{3.7}$$

with $x_{ij} \in \mathbf{R}^4$—containing choke opening, pressures, and temperature—and $q_{ij} \in \mathbf{R}^3$ being the flow rates. From the flow rates we derive the sum of flow, y_{ij}, as in Equation (3.1), and composition, ϕ_{ij}, as in Equation (3.6). Note that the composition, ϕ_{ij}, is available to us in this data set, but is generally an unknown quantity.

With this dataset, the virtual flow meter in Equation (3.2) is viewed as the multitask regression model from Equation (2.22). We use a multitask neural network as our model. Specifically, we use the learned context neural network from Equation (2.41). The inputs to the neural network are the vector of measurements x_{ij} and the composition ϕ_{ij}, represented by γ_{ij} and λ_{ij}, in addition to the *well specific* task parameter vector $\beta_j \in \mathbf{R}^p$. Further, let α denote the set shared neural network parameters required for the layers described by Equation (2.20). The virtual flow meter model is then

$$y_{ij} = f(x_{ij}, \phi_{ij}; \beta_j, \alpha) + e_{ij}. \tag{3.8}$$

In this formulation, we have assumed that both the shared and well-specific parameters are constant. For the shared parameters this is a reasonable assumption because they should capture universal aspects about all wells in all stages of operations. The well-specific parameters are likely to be slightly time-varying, because the state of the system may change, but we make the simplifying assumption that they are constant over the time periods of our data sets.

3.3 Inference problem

In practice, we would use the virtual flow meter in Equation (3.8) to predict the flow rates given a new observation of the measured variables. This would lead us to the posterior predictive distribution from Equation (2.76). For instance, the posterior predictive distribution for an arbitrarily chosen well j, for which we have a new observation of the explanatory variables, $x_{0,j}$, take the form

$$\begin{aligned} p(y_{0,j}, \phi_{0,j}, \beta_j, \alpha | x_{0,j}, \mathcal{D}) = \\ p(y_{0,j} | \phi_{0,j}, \beta_j, \alpha, x_{0,j}, \mathcal{D}) p(\phi_{0,j} | \beta_j, \alpha, x_{0,j}, \mathcal{D}) p(\beta_j, \alpha | x_{0,j}, \mathcal{D}), \end{aligned} \tag{3.9}$$

where $\mathcal{D}$ is the joint data from all wells.

The factors of the posterior predictive distribution can be studied individually. Starting with the third factor in Equation (3.9), which simplifies to $p(\beta_j, \alpha | \mathcal{D})$. This is the posterior distribution of the parameters of the multitask learning model in Equation (3.8). It is based on the assumption that the composition is

known, which is true for the given dataset. A full probabilistic treatment of this multitask problem is extensive. Instead, it is investigated in two steps. Paper I explores a full probabilistic treatment of the single-task version of the problem, using Bayesian neural networks, and Paper II explores a MAP estimate of the multitask formulation. The multitask learning model used in Paper II is studied from a theoretical perspective in Paper III.

The second factor of Equation (3.9), $p(\phi_{0,j}|\beta_j, \alpha, x_{0,j}, \mathcal{D})$, is the problem of estimating the composition over time given the parameters of a virtual flow meter. A variation of this problem is addressed by sequential Monte Carlo in Paper IV.

The first factor in Equation (3.9) simplifies to $p(y_{0,j}|\phi_{0,j}, \beta_j, \alpha, x_{0,j})$ by the same arguments that lead to Equation (2.77). This is the evaluation of our virtual flow meter given all the unknowns. We do not study this distribution any further here. The results of the papers are elaborated in Chapter 5.

Chapter 4

Data and challenges

The majority of the research has been on virtual flow metering using proprietary data. The exception is Paper III, which studies the proposed method on publicly available data from a wide range of domains. These datasets are described in detail in Paper III and are not discussed further here.

This chapter gives a deep dive into the properties of well-test data. We illustrate how well-tests are generated, and highlight statistical learning challenges related to time dependency and correlation between explanatory variables. Additionally, we study how the statistical properties of the data change when all wells are considered simultaneously as opposed to individually. The purpose is to provide some information about the data, as it is not publicly available, and enable the discussions in Chapter 6. All visualization and computations are done with values that are scaled to lie approximately the unit interval.

4.1 Data description

Our data is a collection of well-tests. They consist of the measurements described in Section 3.1. Recall that we consider m wells where choke opening, temperature, and pressures are measured. The flow rates, gas, oil, and water, are measured at a test separator. From the flow rates we derived the gas fraction and oil factor as described by Equation (3.3) and Equation (3.5). Each data points is associated with a timestamp τ_{ij}, and are ordered such that $\tau_{i,j} < \tau_{i+1,j}$.

A well-test is an average value for a set of measurements, taken over a period with stable production. The test period can be detected automatically or set manually. The duration of a well-test is on the scale of hours, while sampling frequency is on the scale of seconds. This means that a typical well-test data point is found as the average of a few thousand samples from each measurement type. The signal-to-noise ratio of these averages is generally quite high. The raw measurements and averaged data for a well-test are illustrated in Figure 4.1. The figure illustrates the gas rate, oil rate, upstream pressure, and choke opening. Downstream pressure, temperature, and water rate are found in the same way. The distance between the location of the well choke valve and the separator can be significant. The measured flow rates are a delayed and smoothed version of the actual flow through the well choke. The exact delay is not known. It is therefore essential to let all measurements involved stabilize before the average values are collected.

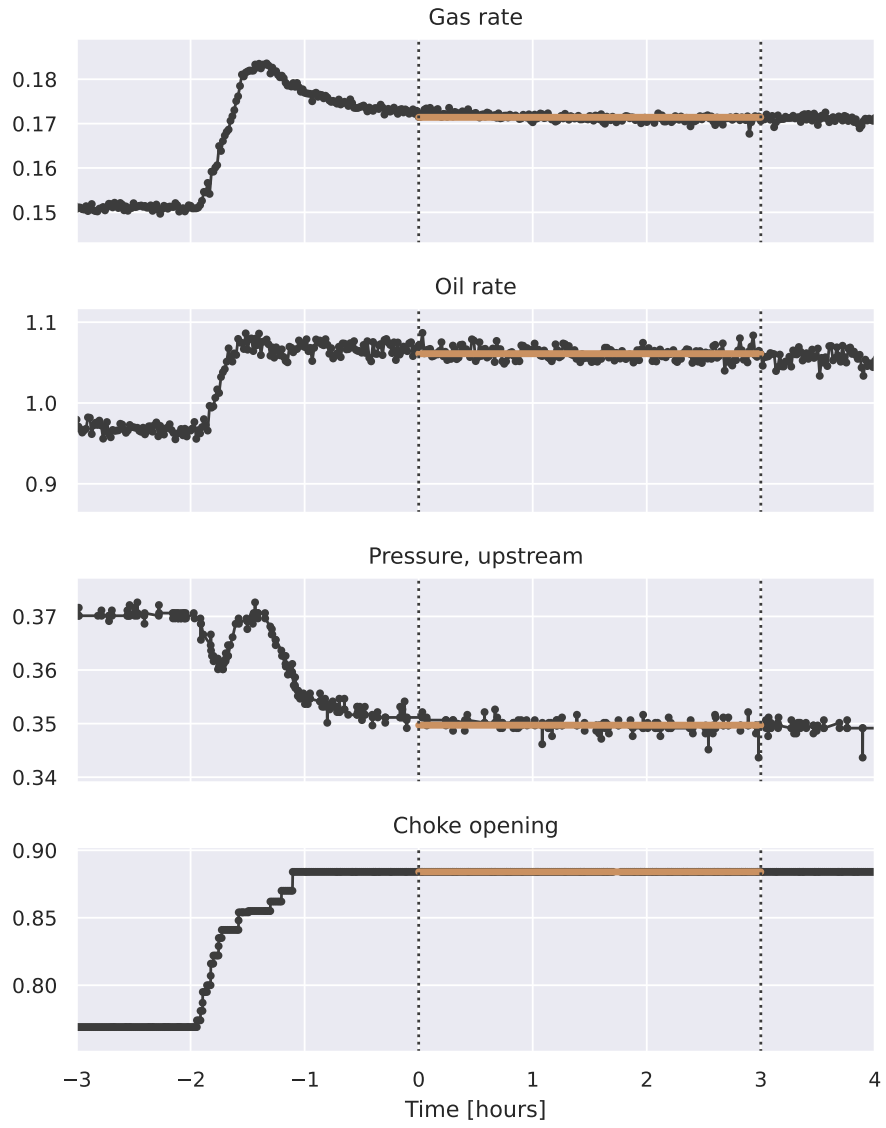


Figure 4.1: Data from a single well-test. Raw data measurements are plotted in black and averaged wells test values are plotted in light brown. well-test values are computed from the samples inside the dotted lines. The well is first routed to the test separator, then the choke is adjusted to reach the desired operating pressure. The well-test period begins when the flow rates and pressures have settled. It last until enough data is collected to ensure a sufficiently accurate average.

4.2 Exploratory analysis

This study in this section is done on a large collection of well-test from a selection of 200 wells. The figures are either examples using a single well that is selected for illustrative purposes, or aggregations of values from all wells. The wells included here have a median of 20 well-tests per year, ranging from less than two to several hundred tests per year.

4.2.1 Time development

The lifetime of a well spans several years, and the collected data is a result of operational practice and reservoir development. The time dependency originates at the reservoir, which is a batch process that generally moves from high pressure to low pressure, and from high oil content to low oil content. Operation of the asset usually aims to produce at maximum capacity for as long as possible, and then attempt to reduce the decline in production as the reservoir is depleted. This leads to measurements being correlated and having time-dependent trends. Additionally, operational practice favors the stability of the process over informative experiments.

The correlations between our variables are summarized in Figure 4.2. We see that for many wells the explanatory variables are either highly correlated or subject to little change. The explanatory variables with the most variation are upstream pressure, choke, and oil factor. These are also highly correlated with each other and have a clear time-dependent trend. This is typical behavior for a well. Downstream pressure and temperature have little variation for many wells. Downstream pressure can be heavily influenced by the separator pressure, which is subject to automatic control and can therefore be almost constant. A benefit of multi-task learning is that the joint data from many tasks can increase the variation in individual signals and reduce the correlations. This is also illustrated in Figure 4.2. We see an increase in variance for all measurements compared to the average well. Measurements with low variance individually see an increase in correlation with other measurements. In particular, the correlation between upstream and downstream pressure becomes significantly stronger when considering data from all wells simultaneously. This is because wells with low upstream pressures must have a low corresponding downstream pressure, while wells with a higher upstream pressure can operate at a broader range of values. Among the most important aspects revealed by Figure 4.2 is that a joint dataset yields a significant reduction in the correlation between choke opening and upstream pressure.

Figure 4.2 indicated a significant time dependency for several explanatory variables. This is explored further in Figure 4.3. For the highlighted well, there is almost no overlap between the upstream pressure values seen in the different years, because the system is constantly moving in the same direction over time. Thus the explanatory variables will drift over time, and predictions made in the future will present a set of explanatory variables that has previously not been observed by the well. When the other wells are superimposed with random start

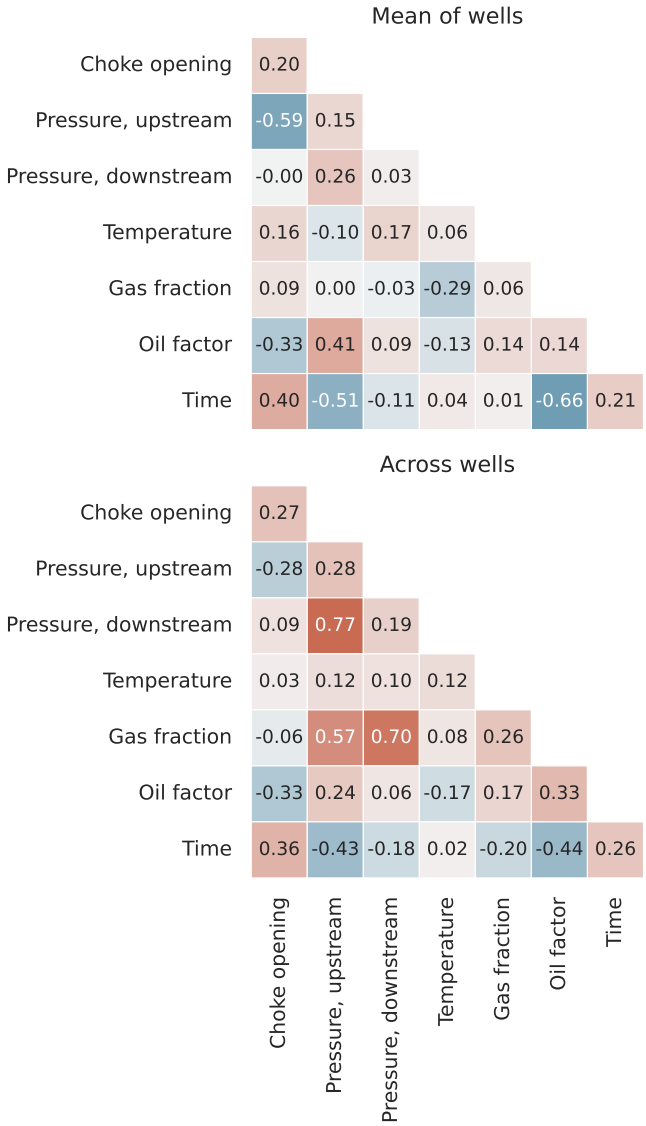


Figure 4.2: Correlation and standard deviation of all measurements and time. Diagonals are the standard deviation of individual measurements. The lower triangle is the correlation between all pairs of measurements. The top plot is the mean of the values found for each well individually. The bottom plot is the values found when merging the data from all wells into one dataset.

times, the picture changes dramatically. The other wells are distributed such that in the years from zero to five there are always multiple wells in all stages of development. This makes the complete set of pressure measurements seen during these years relatively stable, even though no individual well has a stable operation on their own. The set of explanatory variables needed in the future to predict a well may be observed by another well in the past. Thus we will get data coverage in the space of explanatory variables. As long as new wells are appearing regularly, the distribution stays the same. If, hypothetically, all wells started operating at the same time, the benefit of a joint data set would be reduced.

4.2.2 Data splits

Recall our discussion on model generalization and test data in Section 2.1.9. Test data is a subset of our dataset that is set aside during model development in order to fairly assess the models ability to generalize to new data. The properties highlighted in Figure 4.2 and Figure 4.3 indicate that it may be challenging to create a fair and independent test set. This will influence the choices made for model validation. The standard approach is to randomly split the data into training and test sets. This is based on the assumption of independent samples. However, the nature of our problem means that test errors computed using the assumption of independence would be too optimistic for future data. A time-dependent split may give a better indicator of how the model will perform in practice. The difference between a random split and a time-based split is illustrated in Figure 4.4. Here, the frequency of well-tests is high enough to make neighboring samples almost identical. The random split produces a test set that is almost equal to the training data when marginalized to only consider this single measurement. When the split is made by removing the most recent data points, the difference between training and test densities becomes dramatic. In this particular case, there is almost no overlap between the range of training values and test values.

Figure 4.5 compares the two data splitting strategies on all wells. For most of the wells a split based on time yields a greater distance between training and test data than a random split. The effect becomes more pronounced with increased data collection frequency, which is natural when samples close in time are correlated with each other.

4.2.3 Flow composition

As described in Section 2.3, many virtual flow meters make assumptions about the flow composition to estimate the total mass flow. The simplest strategy is to assume the composition is constant between well-tests, but the validity of such assumptions is, of course, highly dependent on the testing frequency. Figure 4.6 illustrates how gas fraction and oil factor develop over several years for one well. The oil factor is steadily decreasing each year. The linear regression line is a good fit, indicating an average change of -0.08 per year. However, while the

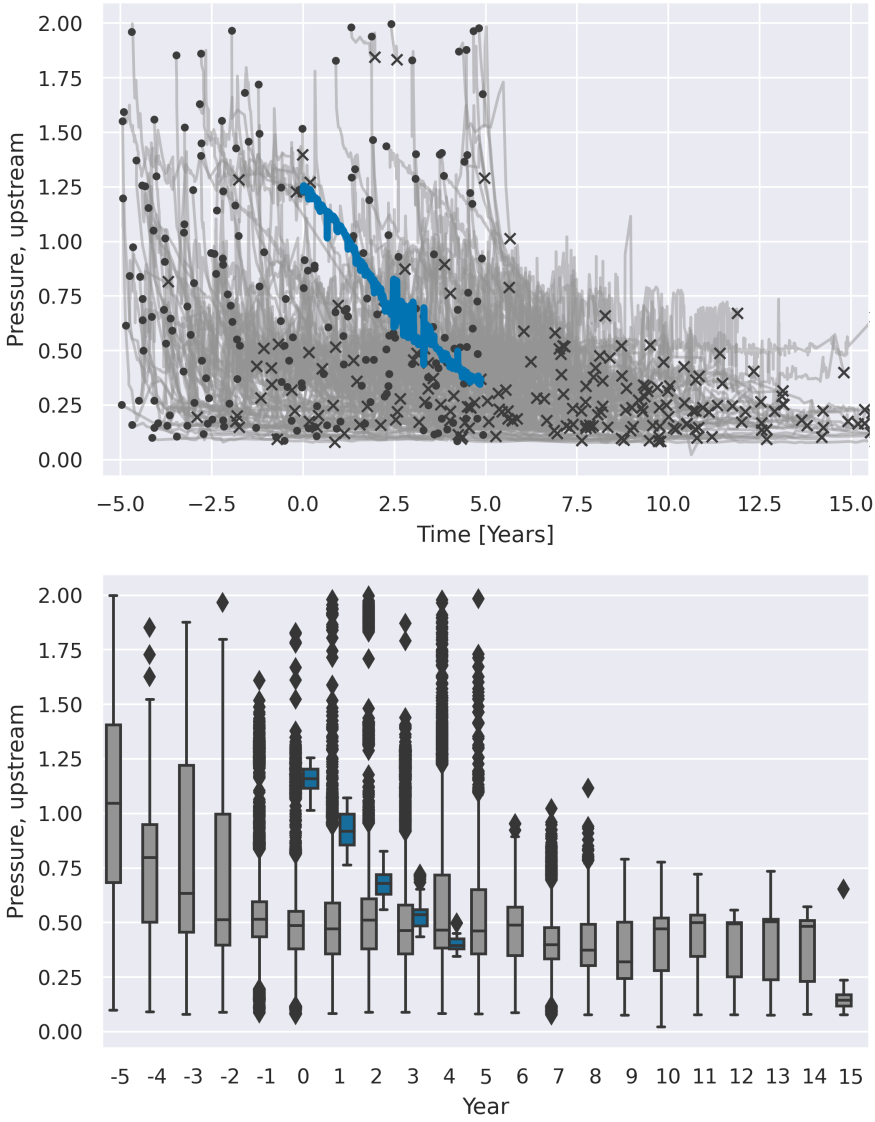


Figure 4.3: Development of upstream pressure over time. A single well is highlighted in blue and compared to the other wells in grey. The highlighted well is given a start time at year zero, while the other wells are given random start times in the range of -5–5. The top plot illustrates the data as time series, where the first data point from each well is a back dot and the last is a black cross. The bottom plot is the same data, but as box plots grouped by year.

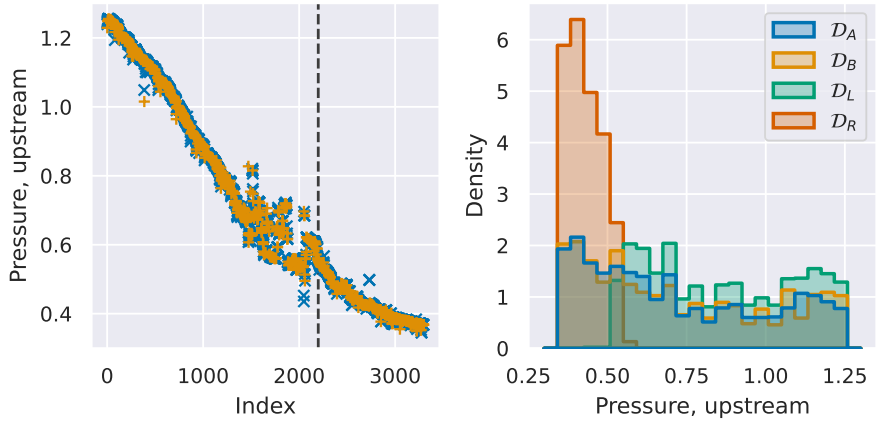


Figure 4.4: Two different data splits illustrated on upstream pressure from a single well. On the left is a scatter plot of the measurements against the datapoint index. On the right is the distribution of values in the different subsets. The sets $\mathcal{D}_A$ and $\mathcal{D}_B$ are a random split of the datapoints, where $\mathcal{D}_A$, plotted in blue, contains 2/3 of the data points and $\mathcal{D}_B$, in orange, contains the remaining 1/3. The other split is on time, marked with a dashed line. $\mathcal{D}_L$ contains the 2/3 of the values to the left of the split and $\mathcal{D}_R$ contains the 1/3 to the right. The quantitative difference between these two splitting procedures is highlighted in Figure 4.5.

long-term trend is clear, there can be significant jumps between neighboring points, indicating the presence of other disturbances that act on top of the reservoir development. For gas fraction, the development is less significant, and the average change over a year is quite small. The changes from one test to the next are on the same level as the oil factor.

Figure 4.7 aggregates the development in compositions for all wells. The same patterns are present for many of the wells. The average developments are quite similar to the well used as an example in Figure 4.6, but the spread is significant, with some wells seeing over twice the rate of change. Gas fraction is less sensitive and many wells see no general trend in its development. It should be noted that this stability in gas fraction can partially be attributed to the use of gas-lift being increased when the reservoir gas is depleted.

4.3 Challenges

We pursue a data-driven virtual flow meter that can provide real-time estimates. The data itself is naturally an essential part of this problem. The overarching theme of our data challenges stems from the desire to keep a naturally evolving

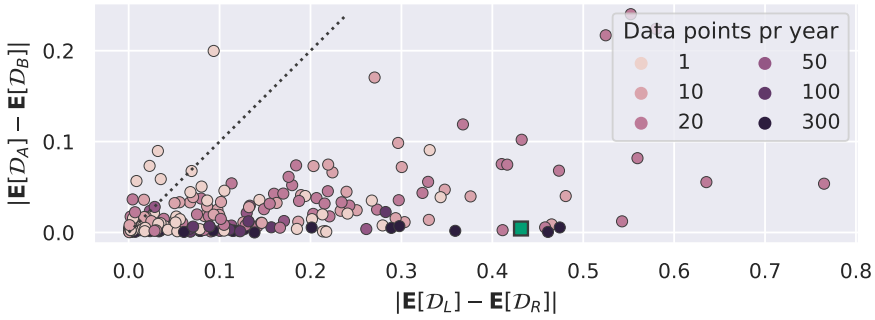


Figure 4.5: Comparison of training and test data distributions for upstream pressure. Consider the two splitting procedures illustrated in Figure 4.4. For both of these splits, the absolute difference in expected value between the training and test data is computed. The differences are plotted against each other and is colored by the number of samples per year, rounded down to the nearest value seen in the legend. The result on the data in Figure 4.4 is marked with a green square.

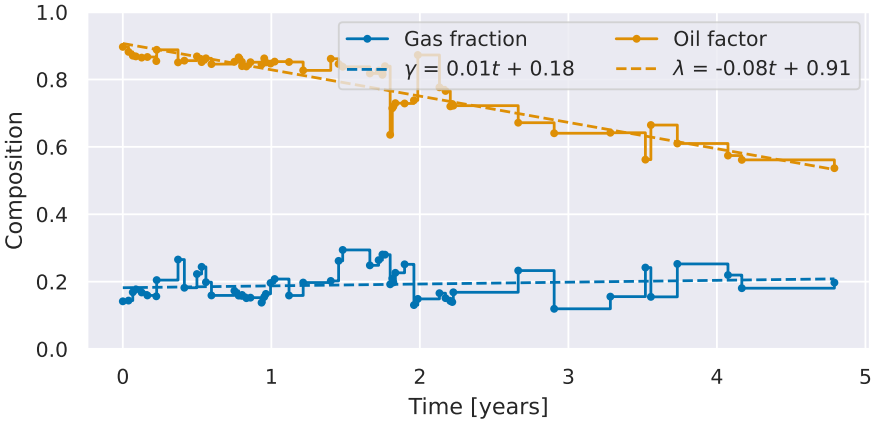


Figure 4.6: Development of gas fraction and oil factor for a single well. The two quantities are given as a scatter plot against time, with a linear regression line superimposed.

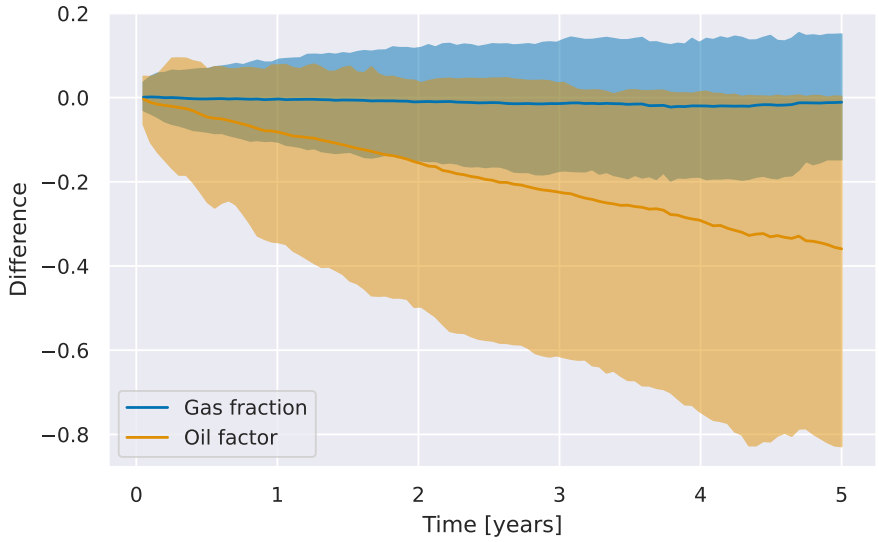


Figure 4.7: Distribution of change in composition as a function of change in time. For each well, the difference in composition is computed for all pairs of well-tests, up to a maximum time span of five years into the future. The results are aggregated and visualized as the mean and 5–95 percentile bands for both quantities.

process as stable as possible. This leads to three main discussion points, lack of excitation, correlated explanatory variables, and a non-stationary process. As discussed in Section 2.2, this is a common problem for soft sensing applications in general.

The desire to keep a process stable results in explanatory variables having low levels of excitation. Downstream pressure is particularly exposed, as it is often closely connected to the operating pressure of the separator facility, which is under automatic control. This means that for many wells the downstream pressure can be nearly constant, as evident from Figure 4.2. As long as the constant explanatory variables *remain* constant, this is not a significant issue, but the models will struggle as soon as operating conditions change.

The desire for stability also leads to a negative correlation between upstream pressure and choke opening. Generally, an increase in these explanatory variables will increase the flow rate. Therefore, as the pressure declines, the choke opening is increased to keep flow rates at a fixed level. Also note that the pressure will change as a function of the choke opening itself, so the relationship between these explanatory variables can be quite complex. This correlation is revealed by Figure 4.2. The correlation makes it challenging to isolate the effects of the

individual explanatory variables with a data-driven model. The joint data set provides a significant reduction in the correlation between these explanatory variables.

The development in the reservoir has a second, and potentially more severe effect. It results in a time-varying distribution of the explanatory variables. Figure 4.3 and Figure 4.5 illustrates the marginal covariate shift in pressure, which generally follows the expected pattern of declining with time. Figure 4.7 illustrates the development of the flow fractions. Both of these are related to the state of the reservoir. These effects are challenging because the wells require models that can be used outside the range of their historical data. Additionally, it makes it hard to quantify the expected performance using the framework outlined in Section 2.1.9.

The asset itself can also change over time. This could be wear and tear, accumulation of solids in the pipeline, or replacement of entire components. Such effects are not directly addressed by the methods proposed here, but a possible future direction is discussed in Chapter 6.

The strategy presented in Chapter 3 relies on multi-task learning to overcome the presented challenges. The joint dataset is able to reduce the severity of these challenges, by increasing variability, changing the correlation structure, and reducing the time dependency of the explanatory variables. This is promising. Other strategies to address these challenges are hybrid models, which introduce known physical relationships to parts of the model. Such models are better suited to extrapolate outside of the training data, making them robust towards the covariate shifts. Additionally, they are likely to have fewer parameters and less flexibility, which makes them able to cope with less excitation in the data.

Chapter 5

Summary of papers

This section gives a brief summary of the four papers produced during this project. They are listed chronologically in the order they were first submitted to their respective journals. Paper I is a large study of the classic single-task neural network virtual flow meter. It provides a baseline for performance and sets the tone for the research direction of this thesis. Paper II innovates on the traditional virtual flow meter architecture by introducing multitask learning. Paper III gives a theoretical deep-dive into the architecture presented in Paper II, and assess its viability in other domains. Paper IV applies Sequential Monte Carlo to estimate time-varying flow composition and tuning parameters for virtual flow meters.

5.1 Paper I: Bayesian neural networks for virtual flow metering: An empirical study

This work provides a large-scale study of data-driven virtual flow meters trained on data from single wells. It explores the use of Bayesian neural networks for this purpose. Additionally, it introduces and discusses a series of data-related challenges that all data-driven virtual flow meters face. Many of these challenges were presented in detail in Chapter 4.

The study compares four single-task variations of the virtual flow meter model in Equation (3.2), all based on the standard feedforward neural network presented in Section 2.1.5. One is a MAP estimate of the parameters, while the other three are probabilistic models based on variational inference. The probabilistic models differ in how the noise terms are modeled.

There are 60 wells used in the study, which is a significantly higher number than what is usually presented in the existing literature. Other works usually contain studies on fewer wells, which allow for a higher degree of tailoring toward each well. We are interested in applying the same architecture to all wells because this would scale better in practice. The consequence is that the models under investigation have excellent performance on some wells and rather poor performance on others.

The performance was tested on both historical and future test data. The four architectures are comparable in their performance on historical data, which was in general quite good for all wells. The performance on future data was significantly worse, with the percentage errors being approximately twice as high. In the future data case, the probabilistic models with a learned noise model had the best performance on the most challenging wells. Additionally, an experiment on the number of training data points illustrated that older data points are less valuable to future predictions.

The performance issues were discussed in the light of identified data challenges and the proposed solutions lead to two directions of research. One approach is hybrid modeling, which attempts to strengthen the model by introducing a combination of derived physical relationships and data-driven elements. The other is to strengthen the model by introducing more data from similar systems. Our focus is on the latter. For the former approach see (Mathilde, Bjarne, and Imsland, 2020).

The calibration of the probabilistic models proved challenging, and it is difficult to set sensible priors. The reason is that neural network parameters do not have any physical interpretation, and are just assumed to be distributed around zero. The width of this distribution essentially becomes a hyper-parameter.

The performance of the Bayesian neural networks where promising, and the possibility of quantifying uncertainty is essential in a real use case. However, in the work that followed we decided to use non-probabilistic neural networks for simplicity. We discuss how the probabilistic networks can be combined with the other results in Chapter 6.

5.2 Paper II: Multi-task learning for virtual flow metering

This work attempts to use multitask learning to address the data challenges identified in Paper I. A multitask neural network architecture is proposed and benchmarked against two traditional single-task learners. The empirical results are based on a study of 55 wells from four assets. Two levels of knowledge transfer were explored, between wells from the same asset and between all wells.

The proposed architecture is composed of two functions, a well-specific domain adaptation, $g : \mathbf{R}^4 \to \mathbf{R}^4$, and a learned context neural network as in Equation (2.41). The composition is

$$y_{ij} = f(g(x_{ij}; \gamma_j), \phi_{ij}; \beta_j, \alpha) + e_{ij}, \tag{5.1}$$

which, apart from the domain adaptation, is equal to the Equation (3.8). In this study, the domain adaptation was used to map the choke opening in percent to a quantity intended to reflect the choke geometry.

Equation (5.1) is composed of two task adaptation mechanisms, the domain adaptation of g and the learned context of the multitask neural network. The impact of the two task adaptation mechanisms was compared, and both were found to be useful. However, in practice, one may favor keeping only the context parameters, as this significantly simplifies the workflow surrounding the model. The context parameters were shown to display sensible behavior in terms of values assigned to wells that are known to be similar and in an analysis of their impact on model predictions. Paper III further explores context parameter architecture with a more theoretical perspective.

The two levels of knowledge sharing, between all wells and between wells from the same asset, were benchmarked against the two single-task learners in an extensive empirical study. The predictive performance was first compared on

a well-by-well basis and then averaged over wells from the same asset. In general, we observed a significant improvement in prediction errors for the two multitask models. However, an element of negative transfer was observed, indicating that not all data was beneficial to all other wells. For instance, one asset performed better without the inclusion of data from the other three assets. The impact of sharing data is the most apparent on the more challenging wells. For the percentiles of wells with low errors, the single-task and multitask neural networks are very close in performance. This indicates that it is hard to improve the performance of a well with a 2% mean absolute prediction error by adding data from more wells. However, for the wells in the percentiles of high errors, the inclusion of more data had a clear benefit. This is in line with the results from Paper I.

The performance of the four models where compared as a function of time since the models were trained. For the first week, the average prediction is often equal to the last seen well-test, which makes all models equally good. As the time since training increased, the benefit of multitask learning became greater. This is related to the non-stationarity of the underlying process, as illustrated in Chapter 4.

In addition to increased predictive performance, the multitask models also see an increased adherence to physical expectations. A study was conducted on the sensitivity of the model toward changes in upstream pressure. It revealed that the multitask learners have a better chance of discovering the expected relationship that an increase in pressure should increase the flow rate.

5.3 Paper III: Multi-task learning by learned context neural networks

This paper is a theoretical and empirical deep-dive into the multitask learning architecture used in Paper II. It isolates the learned context neural network in Equation (5.1), to study the function $y_{ij} = f(x_{ij}; \beta_j, \alpha) + e_{ij}$. The interesting aspect lies in the *learnable* parameters β_j, which allows the neural network to adapt the response to different tasks.

Theoretical properties of the task adaptation mechanism are investigated. Qualitative examples are provided to illustrate how the task parameter interacts with the shared parameters. We proved its flexibility by showing that, theoretically, only one task parameter is needed for adaptation to all tasks. Conversely, the more classical architecture is shown to require one parameter for each task to provide the same guarantee. An empirical study illustrates that in practice the optimal number of task parameters varies with the problem and amount of data.

The performance of the architecture is evaluated on ten publicly available datasets. It is compared to similar architectures, namely the classic multitask neural network of Caruana (1997), given in Equation (2.29), the context-sensitive neural network of Silver, Poirier, and Currie, 2008, given in Equation (2.34), and a linear mixed model, such as in Equation (2.23). The learned-context

architecture is shown to have competitive performance with the other neural network architectures. It seems favorable in cases with few data points, and in situations where few task parameters are desired. The linear model has strong performance on two of the datasets, but the learned context can achieve similar performance despite being heavily over-parametrized for those cases.

In addition to theoretical properties and test performance, the architecture is explored using *hold-out tasks*. A hold-out task is a task that has not been seen during training of the shared parameters. For these tasks, we only train the task parameters using data from the tasks themselves. Results show that identifying parameters for hold-out tasks without updating the shared model is possible given that the properties of the hold-out task are covered by the original tasks.

The study supports the findings in Paper II. The architecture seems fit for the original purpose, but also for similar modeling challenges from other domains. In particular, its ability to manage with few task parameters combined with a well-behaved parameter space facilitates efficient model maintenance. Performance-wise there is still no free lunch, and the compared architectures have individual strengths and weaknesses, but in scenarios with few data points or there is a desire for few task parameters, the proposed architecture is a strong alternative.

5.4 Paper IV: Sequential Monte Carlo applied to virtual flow meter calibration

This paper studies the application of sequential Monte Carlo to calibrate virtual flow meters. The parameters being inferred are the flow composition, using the parametrization from Equation (3.6), and a multiplicative tuning factor. The paper presents a state-space model to describe the evolution of the tuning factor and composition over time and studies its viability in a case with ten wells. The method attempts to utilize data from the production separator, which means that the observed flow rates are a sum of the contributions from, potentially, all ten wells. Well-test data is also available, and the performance of the methods is tested both with and without the inclusion of the well-tests. It reveals that the production data is quite informative and that it is able to provide quite reasonable estimates of the parameters. However, it is also revealed that it is challenging to find a proper balance for the level of noise in the likelihood, and it will likely require significant tuning to achieve satisfactory performance in practice.

The virtual flow meters themselves are taken to be quite simple mechanistic models. However, the state-space model and sampling procedure do not make any assumptions about the virtual flow meter, apart from the ability to be evaluated freely with arbitrary input values. As such, the architectures from Paper I or Paper II could be considered for future experiments.

Chapter 6

Discussion

Data-driven virtual flow metering is not new. The existing literature is extensive and the proposed methods are diverse. Still, there have been very few commercial applications. This led us to investigate the underlying problems, starting with the data itself and the uncertainty of the models. The findings lead us down the path of multitask learning, first for virtual flow meters and then for a broader scope. It culminates in a two-stage strategy for data-driven virtual flow metering. We briefly discuss the findings below.

6.1 Data-driven virtual flow metering

A modeling strategy for data-driven virtual flow metering has been presented. The strategy is comprised of two main components, a static multitask neural network, and a dynamic state space model. The multitask neural network is a mapping from measured and unmeasured state variables to the flow rates through a choke valve. The state-space model provides an estimate of the unknown state. The necessity of a two-stage procedure in combination with multitask learning is due to a continuously changing system with sparse instrumentation.

6.1.1 The available data

Paper I initiated a discussion of challenges related to the data and the system we are trying to model. These challenges are fundamental to the problem itself and will impact any proposed learning strategy. In particular, it concludes that it is unlikely for a single-task learner to generalize to future data with the same performance it achieved on historical data. It also finds that collecting more historical data from the same well has diminishing returns and that it will not amend the problem of a time-varying system. Paper II expands the data study along the same lines. It finds that the explanatory variables are highly correlated and time-varying. Chapter 4 took a deep dive into these data challenges. It explored how changing the dataset to include multiple wells has a beneficial impact on the statistical properties of the data, which can lead to a better-posed learning problem. For instance, the problems caused by each well continuously moving towards lower operating pressures are mitigated by the inclusion of data from other wells that have operated at a broader range of pressures historically. This data was exploited by multitask learning.

6.1.2 multitask learning

Paper II proposes a virtual flow meter based on multitask learning. The underlying idea is that each well is facing the same fundamental modeling

problem and that wells have many aspects in common. A multitask learning formulation allows us to learn model parameters using the joint dataset discussed above. This leads to better performance on future data and better adherence to physical expectations. Increasing the number of wells in the dataset was observed to be beneficial in general, but an element of negative transfer was observed for some wells. The classification of which wells are best learned together is a problem for future research.

The most interesting task adaptation was the learned context parameters, which were studied further in Paper III. However, the domain adaptation mechanism also has its strengths. It can serve as an additional tool to incorporate prior knowledge. While only choke geometry was considered for adaptation in this study, the method could also be used to correct other measurements, such as bias or drift in pressure and temperature sensors.

6.1.3 Probabilistic models

Quantifying uncertainty is an important aspect of building trust in data-driven models. Paper I studied the application of Bayesian neural networks to study the full effect of uncertain network parameters. The method results in reasonably well-calibrated models when testing on historical data. On future test data, the uncertainty assessment becomes more difficult. A full probabilistic treatment of neural networks is quite excessive, and the limited data from individual wells makes it challenging to properly tune these models. It is possible that these shortcomings can be addressed if the Bayesian neural network is combined with the multitask learning approach. Additionally, the mean-field approximation can be problematic, and other alternatives should be investigated. We leave these as directions for future work.

The application of sequential Monte Carlo to estimate tuning factors and composition parameters represents another probabilistic approach explored in this research. The sampling procedure is able to infer reasonable parameters, but we are again faced with the challenging task of finding parameters to properly tune the model. This is particularly challenging when well-test and production data are combined, because well-tests are *usually* very accurate and informative, but can *occasionally* be quite wrong. This makes it difficult to properly describe the uncertainties involved.

6.1.4 Calibration and time varying parameters

All virtual flow meters require some level of maintenance. The model presented in Paper II assumes that the flow composition is known, even though it is changing over time. This will make the virtual flow meter outdated if the composition is not updated. If the frequency of well-testing is high, this is often not a problem. Especially for wells with slow and steady development, such as the case illustrated in Figure 4.6. However, as seen in Figure 4.7, the composition can change quite substantially over a short time frame. To address this, Paper IV

proposed a method based on monitoring the production separator, in an attempt to capture changes in wells early.

While the procedure is able to perform reasonably, it relies heavily on the properties of the asset. For instance, it requires that all wells can be simultaneously modeled and that there are few enough wells that each one has a significant impact on the observed commingled flow. This is not always the case, and alternative calibration strategies should be investigated in future research.

In Paper II it is assumed that each well has a *fixed* task parameter that describes its behavior over the entire period. However, in practice, wells likely change over time. This could be due to wear and tear on the equipment, material accumulating in the pipeline, or maintenance operations. Paper III studied the properties of the task parameters empirically using several different datasets. The findings indicate that the task parameter space is well-behaved, for instance by facilitating the estimation of new tasks that have not been seen during training. This means that it should be possible to incrementally update and adjust the task parameters as new data arrives. This is an opportunity for very efficient workflows regarding model maintenance.

6.1.5 Neural networks

Neural networks were chosen as our modeling framework. The choice was made due to their properties and the ecosystem that surrounds them. Modeling of multi-phase flow results in complex, continuous, nonlinear relationships, and we desired to find these relationships by utilizing large quantities of measurement data. The combination of the learning capacity of neural networks and the benefits of stochastic gradient descent is able to meet this requirement. Neural networks are also easily augmented to multitask learning, and the main innovation in this project relies on the sharing of data between wells. Other methods could have been considered, but evidence points to neural networks being a robust choice for our problem. For instance, Silver, Poirier, and Currie (2008) compares the use of context-sensitive inputs in neural networks, decision trees, and support vector machines. While all three model types are able to train on large data sets to infer non-linear relationships, the empirical study indicates that neural networks are much more capable of utilizing shared information.

There is also a significant interest in neural networks in general. This has led to several mature software packages for implementing and manipulating neural network architectures, which is highly beneficial when conducting the kind of experiments presented here. The continuously increasing understanding of both theoretical and practical aspects of neural networks works to our advantage, and has allowed us to implement and train large and complex models with little effort.

6.1.6 Practical considerations

The three virtual flow meter papers address different aspects of the strategy outlined in Chapter 3. Combining all these aspects into a single model is a

significant challenge. The compartmentalized research allows the results to be incrementally introduced to a real application, which is beneficial from an engineering viewpoint.

The multitask architecture is also beneficial from a maintenance point of view. Paper II briefly discusses the reduction in overall training time, compared to using single-task neural networks for each well individually. Reducing the number of models also reduced the effort required by data scientists to assess and manage the models. Additionally, Paper III provides evidence that the task parameters can be adjusted incrementally over time.

6.2 Generalization to other domains

Paper III explores the learned-context neural network outside of the virtual flow meter domain. The architecture is shown to have beneficial properties when a low-dimensional task parameter space is desired or when there are limited amounts of data. The architecture is therefore likely to be successful in other domains with similar data challenges as those explored here, and it could be a strong contender for other soft-sensor applications.

6.3 Conclusion

Data-driven virtual flow meters face challenges caused by limited data, lack of excitation, and correlated time-dependent explanatory variables. multitask learning takes a significant step to amend these problems, and results in models that generalize better to new data. The presented multitask neural network is also shown to perform well on data from several other domains, which makes it an interesting candidate for soft-sensing applications in general.

However, even with multitask learning, it is essential to keep the models up to date. A calibration procedure based on sequential Monte Carlo was explored for this purpose, but it is unlikely to be a universal solution, and calibration must likely be tailored to individual assets.

The uncertainty of the virtual flow meters can be assessed by Bayesian neural networks, but they were only studied for the single-task case. An exciting future development would be to combine them with multitask learning and explore the potential of learned context Bayesian neural networks.

Bibliography

Balaji, K. et al. (2018). "Status of data-driven methods and their application in oil and gas industry". In: *EAGE Conference and exhibition, Society of Petroleum Engineers*, pp. 1–20.

Balduzzi, D. et al. (2017). "The shattered gradients problem: If resnets are the answer, then what is the question?" In: *International Conference on Machine Learning*. PMLR, pp. 342–350.

Bello, O., Ade-Jacob, S., and Yuan, K. (2014). "Development of hybrid intelligent system for virtual flow metering in production wells". In: *SPE intelligent energy conference & exhibition*. Society of Petroleum Engineers.

Bengio, Y. (2012). "Practical recommendations for gradient-based training of deep architectures". In: *Neural networks: Tricks of the trade*. Springer, pp. 437–478.

Bergstra, J. and Bengio, Y. (2012). "Random search for hyper-parameter optimization". In: *Journal of Machine Learning Research* vol. 13, pp. 281–305.

Bikmukhametov, T. and Jäschke, J. (2019). "Oil Production Monitoring Using Gradient Boosting Machine Learning Algorithm". In: *12th IFAC Symposium on Dynamics and Control of Process Systems, including Biosystems*. Vol. 52. IFAC-PapersOnLine, pp. 514–519.

— (2020). "First Principles and Machine Learning Virtual Flow Metering: A Literature Review". In: *Journal of Petroleum Science and Engineering* vol. 184, no. September 2019, p. 106487.

Blei, D. M., Kucukelbir, A., and McAuliffe, J. D. (2017). "Variational Inference: A Review for Statisticians". In: *Journal of the American Statistical Association* vol. 112, no. 518, pp. 859–877.

Blum, J. R. (1954). "Approximation Methods which Converge with Probability one". In: *The Annals of Mathematical Statistics* vol. 25, no. 2, pp. 382–386.

Bottou, L., Curtis, F. E., and Nocedal, J. (2018). "Optimization methods for large-scale machine learning". In: *SIAM Review* vol. 60, no. 2, pp. 223–311.

Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press.

Caruana, R. (1997). "Multitask Learning". In: *Machine Learning* vol. 28, no. 1, pp. 41–75.

Claeskens, G. and Hjort, N. (2008). *Model Selection and Model Averaging*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press.

Conn, A. R., Scheinberg, K., and Vicente, L. N. (2009). *Introduction to Derivative-free Optimization*. SIAM.

Corneliussen, S. et al. (2005). *Handbook of multiphase flow metering*. 2nd ed. Norwegian Society for Oil and Gas Measurement.

Curreri, F., Patanè, L., and Xibilia, M. G. (2021). "Soft Sensor Transferability: A Survey". In: *Applied Sciences* vol. 11, no. 16.

Cybenko, G. (1989). "Approximation by superpositions of a sigmoidal function". In: *Mathematics of Control, Signals and Systems* vol. 2, no. 4, pp. 303–314.

Demidenko, E. (2004). *Mixed Models: Theory and Applications (Wiley Series in Probability and Statistics)*. USA: Wiley-Interscience.

Doucet, A., De Freitas, N., and Gordon, N. J. (2001). *Sequential Monte Carlo methods in practice*. Springer.

Engelen, J. E. van and Hoos, H. H. (2019). "A survey on semi-supervised learning". In: *Machine Learning* vol. 109, pp. 373–440.

Fedus, W., Zoph, B., and Shazeer, N. (2022). "Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity". In: *Journal of Machine Learning Research* vol. 23, pp. 1–39.

Fortuna, L. et al. (2007). *Soft sensors for monitoring and control of industrial processes*. Vol. 22. Springer.

Foss, B., Knudsen, B. R., and Grimstad, B. (2018). "Petroleum production optimization – A static or dynamic problem?" In: *Computers & Chemical Engineering* vol. 114, pp. 245–253.

Gelman, A., Carlin, J. B., et al. (2013). *Bayesian Data Analysis*. 3rd. Chapman and Hall/CRC.

Gelman, A. and Vehtari, A. (2021). "What are the Most Important Statistical Ideas of the Past 50 Years?" In: *Journal of the American Statistical Association* vol. 116, no. 536, pp. 2087–2097.

Givens, G. H. and Hoeting, J. A. (2012). *Computational statistics*. Vol. 703. John Wiley & Sons.

Gonzaga, J. C. B. et al. (2009). "ANN-based soft-sensor for real-time process monitoring and control of an industrial polymerization process". In: *Computers & Chemical Engineering* vol. 33, no. 1, pp. 43–49.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press.

Grimstad, B. and Sandnes, A. (2016). "Global optimization with spline constraints: a new branch-and-bound method based on B-splines". In: *Journal of Global Optimization* vol. 65, no. 3, pp. 401–439.

Hansen, L. S., Pedersen, S., and Durdevic, P. (2019). "Multi-phase flow metering in offshore oil and gas transportation pipelines: Trends and perspectives". In: *Sensors* vol. 19, no. 9.

Harrou, F. et al. (2021). "A Data-Driven Soft Sensor to Forecast Energy Consumption in Wastewater Treatment Plants: A Case Study". In: *IEEE Sensors Journal* vol. 21, no. 4, pp. 4908–4917.

Hastie, T., Tibshirani, R., and Friedman, J. H. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. 2nd ed. Springer series in statistics. Springer.

He, K. et al. (2015). "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification". In: *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034.

Hospedales, T. et al. (Sept. 2022). “Meta-Learning in Neural Networks: A Survey”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* vol. 44, no. 9, pp. 5149–5169.

Huang, Y. et al. (2023). “Modeling Task Relationships in Multivariate Soft Sensor With Balanced Mixture-of-Experts”. In: *IEEE Transactions on Industrial Informatics* vol. 19, no. 5, pp. 6556–6564.

Hüllermeier, E. and Waegeman, W. (Mar. 2021). “Aleatoric and epistemic uncertainty in machine learning: an introduction to concepts and methods”. In: *Machine Learning* vol. 110, no. 3, pp. 457–506.

Isaksson, A. J., Harjunkoski, I., and Sand, G. (2018). “The impact of digitalization on the future of control and operations”. In: *Computers & Chemical Engineering* vol. 114. FOCAPO/CPC 2017, pp. 122–129.

Jamaluddin, A. K. and Kabir, C. S. (2012). “Flow assurance: Managing flow dynamics and production chemistry”. In: *Journal of Petroleum Science and Engineering* vol. 100, pp. 106–116.

Jansen, J.-D. (2015). *Nodal Analysis of Oil and Gas Wells - Theory and Numerical Implementation*. TU Delft, The Netherlands: Delft University of Technology.

Jiang, Y. et al. (2021). “A Review on Soft Sensors for Monitoring, Control, and Optimization of Industrial Processes”. In: *IEEE Sensors Journal* vol. 21, no. 11, pp. 12868–12881.

Kadlec, P., Gabrys, B., and Strandt, S. (2009). “Data-driven Soft Sensors in the process industry”. In: *Computers & Chemical Engineering* vol. 33, no. 4, pp. 795–814.

Kadlec, P., Grbić, R., and Gabrys, B. (2011). “Review of adaptation mechanisms for data-driven soft sensors”. In: *Computers and Chemical Engineering* vol. 35, no. 1, pp. 1–24.

Kaelbling, L. P., Littman, M. L., and Moore, A. W. (1996). “Reinforcement learning: A survey”. In: *Journal of artificial intelligence research* vol. 4, pp. 237–285.

Kanshio, S. (2020). “A review of hydrocarbon allocation methods in the upstream oil and gas industry”. In: *Journal of Petroleum Science and Engineering* vol. 184.

Kidger, P. and Lyons, T. (2020). “Universal Approximation with Deep Narrow Networks”. In: *Proceedings of Thirty Third Conference on Learning Theory*. Vol. 125. Proceedings of Machine Learning Research, pp. 2306–2327.

King, D. E. (2009). “Dlib-ml: A machine learning toolkit”. In: *Journal of Machine Learning Research* vol. 10, pp. 1755–1758.

LeCun, Y. et al. (1989). “Backpropagation Applied to Handwritten Zip Code Recognition”. In: *Neural Computation* vol. 1, no. 4, pp. 541–551.

Lu, H. et al. (2019). “Oil and Gas 4.0 era: A systematic review and outlook”. In: *Computers in Industry* vol. 111, pp. 68–90.

Malherbe, C. and Vayatis, N. (2017). “Global optimization of Lipschitz functions”. In: *Proceedings of the 34th International Conference on Machine Learning*. Vol. 70. Proceedings of Machine Learning Research. Sydney, NSW, Australia, pp. 2314–2323.

Maschler, B. and Weyrich, M. (2021). “Deep Transfer Learning for Industrial Automation: A Review and Discussion of New Techniques for Data-Driven Machine Learning”. In: *IEEE Industrial Electronics Magazine* vol. 15, no. 2, pp. 65–75.

Mathilde, H., Bjarne, G., and Imsland, L. (2020). “Developing a Hybrid Data-Driven, Mechanistic Virtual Flow Meter - a Case Study”. In: *IFAC-PapersOnLine* vol. 53, no. 2. 21th IFAC World Congress, pp. 11692–11697.

Monteiro, D. D. et al. (2020). “Using Data analytics to quantify the impact of production test uncertainty on oil flow rate forecast”. In: *IFP Energies Nouvelles* vol. 75 (7), pp. 1–15.

Nalisnick, E., Smyth, P., and Tran, D. (2023). “A Brief Tour of Deep Learning from a Statistical Perspective”. In: *Annual Review of Statistics and Its Application* vol. 10, no. 1, pp. 219–246.

Nocedal, J. and Wright, S. J. (2006). *Numerical Optimization*. Second Edition. Springer.

Ovadia, Y. et al. (2019). “Can You Trust Your Model’s Uncertainty? Evaluating Predictive Uncertainty Under Dataset Shift”. In: *33rd Conference on Neural Information Processing Systems*. Vol. 32.

Pan, L. et al. (2023). “TMoE-P: Towards the Pareto Optimum for Multivariate Soft Sensors”. In: pp. 1–13. arXiv: 2302.10477.

Paszke, A. et al. (2019). “PyTorch: An Imperative Style, High-Performance Deep Learning Library”. In: *Advances in Neural Information Processing Systems* vol. 32, pp. 8026–8037.

Qiao, J., Zhou, J., and Meng, X. (2023). “A Multitask Learning Model for the Prediction of NOx Emissions in Municipal Solid Waste Incineration Processes”. In: *IEEE Transactions on Instrumentation and Measurement* vol. 72, no. x, pp. 1–14.

Quinonero-Candela, J. et al. (2009). *Dataset shift in machine learning*. Mit Press.

AL-Qutami, T. A. et al. (2017). “Development of soft sensor to estimate multiphase flow rates using neural networks and early stopping”. In: *International Journal on Smart Sensing and Intelligent Systems* vol. 10, no. 1, pp. 199–222.

— (2018). “Virtual multiphase flow metering using diverse neural network ensemble and adaptive simulated annealing”. In: *Expert Systems with Applications* vol. 93, pp. 72–85.

Raudenbush, S. W. and Bryk, A. S. (2002). *Hierarchical linear models: Applications and data analysis methods*. Vol. 1. Sage.

Robbins, H. and Monro, S. (1951). “A Stochastic Approximation Method”. In: *The Annals of Mathematical Statistics* vol. 22, no. 3, pp. 400–407.

Schmidhuber, J. (2015). “Deep learning in neural networks: An overview”. In: *Neural networks* vol. 61, pp. 85–117.

Shaban, H. and Tavoularis, S. (2014). “Measurement of gas and liquid flow rates in two-phase pipe flows by the application of machine learning techniques to differential pressure signals”. In: *International Journal of Multiphase Flow* vol. 67, pp. 106–117.

Shi, X. et al. (2021). “A synchronous prediction model based on multi-channel cnn with moving window for coal and electricity consumption in cement calcination process”. In: *Sensors* vol. 21, no. 13.

Silver, D. L., Poirier, R., and Currie, D. (2008). “Inductive transfer with context-sensitive neural networks”. In: *Machine Learning* vol. 73, no. 3, pp. 313–336.

Solle, D. et al. (2017). “Between the Poles of Data-Driven and Mechanistic Modeling for Process Operation”. In: *Chemie Ingenieur Technik* vol. 89, no. 5, pp. 542–561.

Sun, Q. and Ge, Z. (2021). “A Survey on Deep Learning for Data-Driven Soft Sensors”. In: *IEEE Transactions on Industrial Informatics* vol. 17, no. 9, pp. 5853–5866.

Thorn, R., Johansen, G. A., and Hjertaker, B. T. (2012). “Three-phase flow measurement in the petroleum industry”. In: *Measurement Science and Technology* vol. 24, no. 1.

Toskey, E. (2012). “Improvements to Deepwater Subsea Measurements RPSEA Program: Evaluation of Flow Modeling”. In: *Proceedings of the Annual Offshore Technology Conference*, pp. 1–18.

Yan, S. and Yan, X. (2020). “Joint monitoring of multiple quality-related indicators in nonlinear processes based on multi-task learning”. In: *Measurement: Journal of the International Measurement Confederation* vol. 165, p. 108158.

Yuksel, S. E., Wilson, J. N., and Gader, P. D. (2012). “Twenty years of mixture of experts”. In: *IEEE Transactions on Neural Networks and Learning Systems* vol. 23, no. 8, pp. 1177–1193.

Zhai, N. et al. (2023). “Soft Sensor Model for Billet Temperature in Multiple Heating Furnaces Based on Transfer Learning”. In: *IEEE Transactions on Instrumentation and Measurement* vol. 72, pp. 1–13.

Zhang, Y. and Yang, Q. (2021). “A Survey on Multi-Task Learning”. In: *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–20.

Zintgraf, L. et al. (2019). “Fast Context Adaptation via Meta-Learning”. In: *Proceedings of the 36th International Conference on Machine Learning*. Vol. 97. Proceedings of Machine Learning Research, pp. 7693–7702.

Papers

Paper I

I

Bayesian neural networks for virtual flow metering: An empirical study

Bjarne Grimstad, Mathilde Hotvedt, Anders T. Sandnes, Odd Kolbjørnsen, Lars S. Imsland

Published in *Applied Soft Computing*, Volume 112, November 2021, 107776, DOI: 10.1016/j.asoc.2021.107776.

Abstract

Recent works have presented promising results from the application of machine learning (ML) to the modeling of flow rates in oil and gas wells. Encouraging results and advantageous properties of ML models, such as computationally cheap evaluation and ease of calibration to new data, have sparked optimism for the development of data-driven virtual flow meters (VFMs). Data-driven VFMs are developed in the small data regime, where it is important to question the uncertainty and robustness of models. The modeling of uncertainty may help to build trust in models, which is a prerequisite for industrial applications. The contribution of this paper is the introduction of a probabilistic VFM based on Bayesian neural networks. Uncertainty in the model and measurements is described, and the paper shows how to perform approximate Bayesian inference using variational inference. The method is studied by modeling on a large and heterogeneous dataset, consisting of 60 wells across five different oil and gas assets. The predictive performance is analyzed on historical and future test data, where an average error of 4-6% and 8-13% is achieved for the 50% best performing models, respectively. Variational inference appears to provide more robust predictions than the reference approach on future data. Prediction performance and uncertainty calibration is explored in detail and discussed in light of four data challenges. The findings motivate the development of alternative strategies to improve the robustness of data-driven VFMs.

I.1 Introduction

Knowledge of multiphase flow rates is essential to efficiently operate a petroleum production asset. Measured or predicted flow rates provide situational awareness

and flow assurance, enable production optimization, and improve reservoir management and planning. However, multiphase flow rates are challenging to obtain with great accuracy due to uncertain subsurface fluid properties and complex multiphase flow dynamics (Jansen, 2015). In most production systems, flow rates are measured using well testing. While these measurements are of high accuracy, they are intermittent and infrequent (Monteiro, Duque, et al., 2020). Some production systems have multiphase flow meters (MPFMs) installed at strategic locations to continuously measure flow rates. Yet, these devices are expensive, and typically have lower accuracy than well testing. An alternative approach is to compute flow rates using virtual flow metering (VFM). VFM is a soft-sensing technology that infers the flow rates in the production system using mathematical models and ancillary measurements (Toskey, 2012). Many fields today use some form of VFM technology complementary to flow rate measurements. There are two main applications of a VFM: i) real-time prediction of flow rates, and ii) prediction of historical flow rates. The second application is relevant to the prediction of missing measurements due to sensor failure or lacking measurements in between well tests.

VFMs are commonly labeled based on their use of either mechanistic or data-driven models (Bikmukhametov and Jäschke, 2020). Both model types can be either dynamic or steady-state models. Mechanistic VFM models are derived from prior knowledge about the internal structure of the process (Solle et al., 2017). Physical, first-principle laws such as mass, energy, and momentum-balance equations, along with empirical closure relations, are utilized to describe the relationship between the system variables. Mechanistic modeling is the most common approach in today's industry and some commercial VFMs are Prosper, ValiPerformance, LedaFlow, FlowManager, and Olga (Amin, 2015).

In contrary to mechanistic models, data-driven models exploit patterns in process data and attempt to find relationships between the system variables with generic mathematical models. In other words, data-driven models attempt to model the process without utilizing explicit prior knowledge (Solle et al., 2017). In recent years, there has been an increasing number of publications on data-driven VFMs (Bikmukhametov and Jäschke, 2020). The developments are motivated by the increasing amount of sensor data due to improved instrumentation of petroleum fields, better data availability, more computing power, better machine learning tools and more practitioners (Balaji et al., 2018). Additionally, data-driven VFMs may require less maintenance than a mechanistic VFMs (AL-Qutami, Ibrahim, Ismail, and Ishak, 2017c). Even so, commercial data-driven VFMs are rare. This is arguably due to the following data challenges, which must be overcome by data-driven VFMs:

1. Low data volume
2. Low data variety
3. Poor measurement quality
4. Non-stationarity of the underlying process

The first two challenges are due to data-driven methods, especially neural networks, being data-hungry, and require substantial data volume and variety to achieve high accuracy (Mishra and Datta-Gupta, 2018). Petroleum production data do not generally fulfill these requirements. For petroleum fields without continuous monitoring of the flow rates, new data is obtained at most 1-2 times per month during well testing (Monteiro, Duque, et al., 2020), yielding low data volume. For fields with continuous measurements, the data volume may be higher, yet, the second challenge of low variety remains. Low data variety relates to the way production systems are operated and how it affects the information content in historical production data. The production from a well is often kept fairly constant by the operator, in particular during plateau production, i.e., when the production rate is limited by surface conditions such as the processing capacity. When a field later enters the phase of production decline, the operator compensates for falling pressures and production rates by gradually opening the production choke valves. This can introduce correlations among the measured variables which are unfortunate for data-driven models. A common consequence of modeling in the small data regime is overfitting which decreases the generalization ability of the model, that is, the models struggle with extrapolation to unseen operating conditions (Solle et al., 2017). Nonetheless, one should be able to model the dominant behavior of the well and make meaningful predictions close to the observed data if care is taken to prevent overfitting (AL-Qutami, Ibrahim, and Ismail, 2018).

The third challenge, poor measurement quality, highly influences the predictive abilities of data-driven VFMs. Common issues with measurement devices in petroleum wells include measurement noise, bias, and drift. Additionally, equipment or communication failures may lead to temporarily or permanently missing data. Common practices to improve data quality include device calibration, data preprocessing and data reconciliation (Câmara et al., 2017). In model development, methods such as parameter regularization and model selection techniques prevent overfitting of the model in the presence of noisy data. However, some of the above issues and practices may be challenging to handle in a data-driven model.

Lastly, the underlying process in petroleum production systems, the reservoir, is non-stationary. The pressure in the reservoir decreases as the reservoir is drained and the composition of the produced fluid changes with time (Foss, Knudsen, and Grimstad, 2018). Time-varying boundary conditions make it more difficult to predict future process behavior for data-driven VFMs as they often struggle with extrapolation. As mentioned above, methods to prevent overfitting to the training data in model development may (and should) be utilized to improve extrapolation abilities to the near future, and frequent model updating or online learning would contribute to better predictive abilities for larger changes in the underlying process.

As the above discussion reflects, data-driven VFMs are influenced by uncertainty. Both model (epistemic) uncertainty and measurement (aleatoric) uncertainty are present (Hüllermeier and Waegeman, 2021). The first type originates from the model not being a perfect realization of the true process

and there are uncertainties related to the model structure and parameters. The latter type is a cause of noisy data due to imprecision in measurements (Gal, 2016). Accounting for uncertainty is important to petroleum production engineers as they are often concerned with worst- and best-case scenarios. Further, information about the prediction uncertainty may aid the production engineers to decide whether the model predictions may be trusted. According to a recent survey (Bikmukhametov and Jäschke, 2020), uncertainty estimation must be addressed by future research on VFM.

The motivation of this paper is to address uncertainty by introducing a probabilistic, data-driven VFM based on Bayesian neural networks. With this approach, epistemic uncertainty is modeled by considering the weights and biases of the neural network as random variables. Aleatoric uncertainty can be accommodated by a homoscedastic or heteroscedastic model of the measurement noise. This allows the modeler to separately specify priors related to the two uncertainty types. This can be beneficial when having knowledge of the measurement devices that produced the data modeled on.

Historically, the difficulty of performing Bayesian inference with neural networks has been a hurdle to practitioners. We thus provide a description of how to train the model using variational inference. Variational inference provides the means to perform efficient, approximate Bayesian inference and results in a posterior distribution over the model parameters (Blei, Kucukelbir, and McAuliffe, 2017). The method has shown promising results in terms of quantifying prediction uncertainty on other problems subject to small datasets and dataset shift (Ovadia et al., 2019). We also consider maximum a posteriori estimation, which serves as a non-probabilistic reference method. Although it computes a point estimate of the parameters, as opposed to a posterior distribution, it more closely resembles the maximum likelihood methods used in the majority of previous works on data-driven VFM. The reference method enables us to investigate if a probabilistic method, i.e. variational inference, may improve robustness over a non-probabilistic method. We test the proposed VFM by performing a large-scale empirical study on data from a diverse set of 60 petroleum wells.

The paper is organized as follows. In Section I.2 we briefly survey related works on data-driven VFM, with a focus on applications of neural networks. This section also gives some relevant background on probabilistic modeling. In Section I.3 we describe how flow rates are measured and the dataset used in the case study. The probabilistic model for data-driven VFM is presented in Section I.4 and in Section I.5 we discuss methods for Bayesian inference. The case study is presented in Section I.6 and discussed in Section I.7. In Section I.8 we conclude and give our recommendations for future research on data-driven VFM based on our findings.

I.2 Related work

I.2.1 Traditional data-driven modeling

In literature, several data-driven methods have been proposed for VFM modeling, for instance, linear and nonlinear regression, principal component regression, random forest, support vector machines and the gradient boosting machine learning algorithm (Bello, Ade-Jacob, and Yuan, 2014; Bikmukhametov and Jäschke, 2019; Xu et al., 2011; Zangl, Hermann, and Christian, 2017). One of the most popular and promising data-driven methods for VFM are neural networks (NN). In Zangl, Hermann, and Christian (2017), the oil flow rate from three wells was modeled using NNs, and an error as low as 0.15% was reported. However, well-step tests were used to generate data with sufficient variety, and the time-span of the data covered only 30 hours. The three studies, Ahmadi et al. (2013), S. M. Berneti and Shahbazian (2011), and Hasanvand and S. Berneti (2015), investigated NNs for the oil flow rate from a reservoir using data samples from 31-50 wells. All used a neural network architecture with one hidden layer and 7 hidden neurons. In the two first, the imperialist competitive algorithm was used to find the NN weights. All of the three studies reported a very small mean squared error, of less than 0.05. Yet, the data was limited to a time-span of 3 months and did not include measurements of the choke openings of the petroleum wells. This will strongly affect the future model performance when reservoir conditions change and the choke openings are adjusted.

A particularly noticeable series of studies on VFM and NN, using historical well measurements with a time-span of more than a year, are AL-Qutami, Ibrahim, and Ismail (2018), AL-Qutami, Ibrahim, Ismail, and Ishak (2017a), AL-Qutami, Ibrahim, Ismail, and Ishak (2017b), and AL-Qutami, Ibrahim, Ismail, and Ishak (2017c). In AL-Qutami, Ibrahim, Ismail, and Ishak (2017a), the oil and gas flow rates were modeled using two individual feed-forward NN, with one hidden layer and 6 and 7 neurons respectively, and with early stopping to prevent overfitting. An error of 4.2% and 2.3% for the oil and gas flow rates were reported. In AL-Qutami, Ibrahim, Ismail, and Ishak (2017c), a radial basis function network was utilized to model the gas flow rate from four gas condensate wells, and the Orthogonal Least Squares algorithm was applied to find the optimal number of neurons (≤ 80) in the hidden layer of the network. The study reported an error of 5.9%. In AL-Qutami, Ibrahim, and Ismail (2018) and AL-Qutami, Ibrahim, Ismail, and Ishak (2017b), ensemble neural networks were used to excel the learning from sparse data. In the first, the neural network architecture was limited to one hidden layer but the number of hidden neurons was randomly chosen in the range 3-15. Errors of 1.5%, 6.5%, and 4.7% for gas, oil, and water flow rate predictions were achieved. The second paper considered 1-2 hidden layers with 1-25 neurons. Errors of 4.7% and 2.4% were obtained for liquid and gas flow rates respectively.

I.2.2 Probabilistic modeling

A common approach in today's industry and literature is to study the sensitivity of the model to changes in parameter values, thus to a certain extent approaching epistemic uncertainty, e.g. Bieker, Slupphaug, and Johansen (2007), Fonseca, Gonçalves, and Azevedo (2009), Monteiro, Chaves, et al. (2017), Monteiro, Duque, et al. (2020), and Zangl, Hermann, and Christian (2017). By approximating probability distributions for some of the model parameters from available process data and using sampling methods to propagate realizations of the parameters through the model, a predictive distribution of the output with respect to the uncertainty in the parameter may be analyzed.

Probabilistic modeling offers a more principled way to model uncertainty, e.g. by considering model parameters and measurement noise as random variables (Ghahramani, 2015). With Bayesian inference, a posterior distribution of the model output is found that takes into account both observed process data and prior beliefs of the model parameters (Hastie, Tibshirani, and Friedman, 2009). The result is a predictive model that averages over all likely models that fit the data and a model that offers a natural parameter regularization scheme through the use of priors. This is in contrast to traditional data-driven modeling where the concern is often to find the maximum likelihood estimate (Ghahramani, 2015). Although probabilistic models and Bayesian inference are well-known in other fields of research, probabilistic VFMs are rare, yet existent (Bassamzadeh and Ghanem, 2018; Lorentzen, Stordal, Luo, et al., 2016; Lorentzen, Stordal, Nævdal, et al., 2014; Luo et al., 2014).

The following series of studies, Lorentzen, Stordal, Luo, et al. (2016), Lorentzen, Stordal, Nævdal, et al. (2014), and Luo et al. (2014), constructed a mechanistic, probabilistic model of the flow rate in petroleum wellbores. A method for probabilistic, data-driven models is Bayesian neural networks (BNNs). BNNs are similar to traditional neural networks but with each parameter represented with a probability distribution (Hastie, Tibshirani, and Friedman, 2009; Polson and Sokolov, 2017). Bayesian methods have shown to be efficient in finding high accuracy predictors in small data regimes and in the presence of measurement noise without overfitting to the data (Snoek, Larochelle, and Adams, 2012). Further, Bayesian methods lend themselves to online model updating and could quickly improve the model's predictive ability when introduced to new operating regions. Yet, there are disadvantages with probabilistic modeling and Bayesian inference. Except in special cases, inferring the posterior probability distribution of the model consists of solving intractable integrals and inference is slow for large datasets (Blei, Kucukelbir, and McAuliffe, 2017). However, methods for approximation of the posterior distribution exist such as Markov Chain Monte Carlo (MCMC) sampling and variational inference (VI). Comparing these two approximation methods, VI has shown to scale better to large datasets and inference tends to be faster. Additionally, it simplifies posterior updating in the presence of new data. Nevertheless, the approximation with VI is in most cases bounded away from the true distribution, whereas MCMC methods will in principle converge towards the true distribution (Blei, Kucukelbir, and McAuliffe,

2017). A challenge for data-driven probabilistic models, such as Bayesian neural networks, is that the model parameters are generally non-physical, and setting the parameter priors is nontrivial. Despite neural networks being among the more popular data-driven methods for VFM modeling, to the extent of the authors' knowledge, there has been no attempt at using BNNs for VFM. There are, however, examples of BNNs being used for data-driven prediction in similar applications (Humphrey et al., 2016; Liu et al., 2012).

I.3 Flow rate measurements and dataset

A petroleum production well is illustrated in Figure I.1. Produced fluids flow from the reservoir, up to the wellhead, and through the choke valve. The choke valve opening (u) is operated to control the production from the well. The fluids thereafter enter the separator which separates the multiphase flow into the three single phases of oil, gas, and water $\boldsymbol{q} = (q_{\text{oil}}, q_{\text{gas}}, q_{\text{wat}})$. On well-instrumented wells, pressure (p) and temperature (T) is measured upstream and downstream the choke valve.

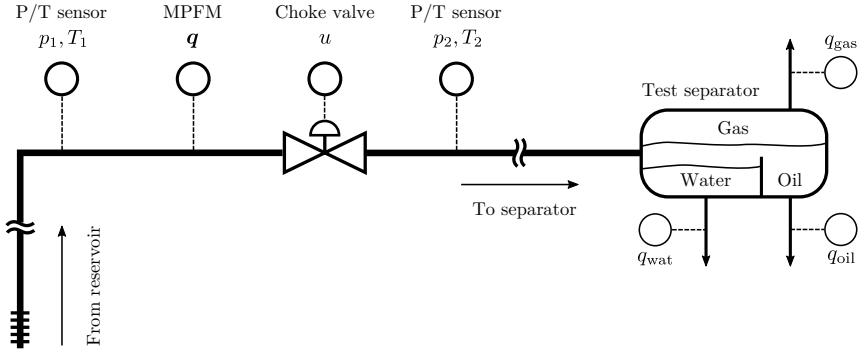


Figure I.1: Sensor placement in a typical production well. A MPFM measures multiphase flow rates in the well. During well testing, single phase flow rates are measured with high accuracy after fluid separation.

The two main devices to measure multiphase flow rates in a well are the multiphase flow meter (MPFM) and test separator, both illustrated in Figure I.1. MPFMs are complex devices based on several measurement principles and offer continuous measurements of the multiphase flow rate. Unfortunately, MPFMs have limited operation range, struggle with complex flow patterns, and are subject to drift over time (Corneliussen et al., 2005). Additionally, PVT (pressure-volume-temperature) data are used as part of the MPFM calculations and should be accurate and up-to-date for high accuracy MPFM measurements. On the other hand, well-testing is performed by routing the multiphase flow to

a test separator whereby the separated flows are measured using single-phase measurement devices over a period of time (typically a few hours). Compared to the MPFM, well tests are performed infrequently, usually 1-2 times a month (Monteiro, Duque, et al., 2020).

Normally, measurements of the multiphase flow rate obtained through well-testing have higher accuracy than the measurements from the MPFMs. This is due to the use of single-phase measurement devices in well-testing. According to Corneliussen et al. (2005) and Marshall and Thomas (2015), the uncertainty, in terms of mean absolute percentage error, of well tests, may potentially be as low as 2% and 1% for gas and oil respectively, whereas MPFM uncertainty is often reported to be around 10%. The error statistics are calculated with respect to reference measurements. For measurements of pressure, temperature, and choke openings, we assume that the sensors' accuracy is high, typically with an uncertainty of 1% or less, and measurement error in these measurements are therefore neglected.

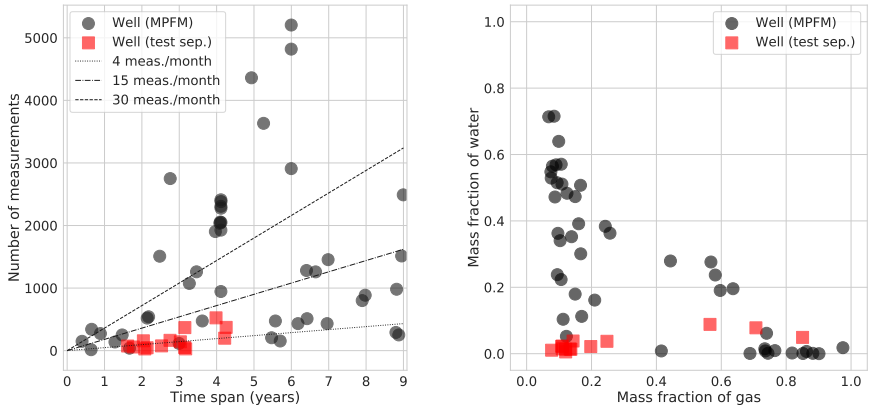
The flow rates are often given as volumetric flow rates under standard conditions, e.g. as standard cubic meter per hour (Sm^3/h). Standard conditions make it easier to compare to reference measurements or measurements at other locations in the process as the volume of the fluid changes with pressure and temperature. Flow rates may be converted from actual conditions to standard conditions using PVT data (Krejbjerg et al., 2019). If the density of the fluid at standard conditions is known, the standard volumetric flow rate may be converted to mass flow rate, and the phasic mass fractions, $\boldsymbol{\eta} = (\eta_{\text{oil}}, \eta_{\text{gas}}, \eta_{\text{wat}})$, may be calculated. We assume steady-state production, frozen flow, and incompressible liquid such that the phasic volumetric flow rate and mass fractions are constant through the system, from the reservoir to the separator.

I.3.1 Dataset

The dataset used in this study consists of 66 367 data points from 60 wells producing from five oil and gas fields. The dataset was produced from raw measurement data using a data squashing technology (Grimstad et al., 2016). The squashing procedure averages raw measurement data in periods of steady-state operation to avoid short-scale instabilities. The resulting data points, which we refer to as measurements henceforth, are suitable for modeling of steady-state production rates.

For each well we have a sequence of measurements in time. The time span from the first to last measurement is plotted for each well in Figure I.2a. The figure shows that the measurement frequency varies from a handful to hundreds of measurements per year. There are 14 wells with test separator measurements, for which the average number of measurements is 163. The other 46 wells have MPFM measurements, and the average number of measurements is 1393. The 60 wells are quite different from each other in terms of produced fluids. Figure I.2b illustrates the spread in mass fractions among the wells.

In the following, we model the multiphase flow through the production choke valve, a crucial component in any VFM. We consider ideal conditions,



(a) Measurement frequency

(b) Average mass fractions

Figure I.2: The number of measurements is plotted against the time span from the first to last measurement in (a). The average gas and water mass fraction is shown for all wells in (b).

in the sense that all measurements required by a reasonable choke model are available (Mathilde, Bjarne, and Imsland, 2020). For each well, we collect the corresponding measurements in a dataset $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$. We will only consider one well at the time and simply refer to the dataset as $\mathcal{D}$. The target variable is the total volumetric flow rate, $y_i = q_{\text{oil},i} + q_{\text{gas},i} + q_{\text{wat},i} \in \mathbb{R}$, measured *either* by a test separator *or* a MPFM. The explanatory variables,

$$\boldsymbol{x}_i = (u_i, p_{1,i}, p_{2,i}, T_{1,i}, T_{2,i}, \eta_{\text{oil},i}, \eta_{\text{gas},i}) \in \mathbb{R}^7,$$

are the measured choke opening, the pressures and temperatures upstream and downstream the choke valve, and the mass fractions of oil and gas. No experimental set-up was used to affect the data variety; for example, we did not consider step well tests as in Zangl, Hermann, and Christian (2017).

I.4 Probabilistic flow model

Consider the following probabilistic model for the total multiphase flow rate:

$$\left.\begin{aligned} y_i &= z_i + \epsilon_i \\ z_i &= f(\boldsymbol{x}_i, \boldsymbol{\phi}) \\ s_i &= g(z_i, \boldsymbol{\psi}) \\ \epsilon_i &\sim \mathcal{N}(0, s_i^2) \end{aligned}\right\} i = 1, \ldots, N, \\ \boldsymbol{\phi} \sim p(\boldsymbol{\phi}) = \prod_{i=1}^{K_\phi} \mathcal{N}(\phi_i \,|\, a_i, b_i^2), \\ \boldsymbol{\psi} \sim p(\boldsymbol{\psi}) = \prod_{i=1}^{K_\psi} \mathcal{N}(\psi_i \,|\, c_i, d_i^2), \tag{I.1}$$

where y_i is a measurement of the multiphase flow rate z_i subject to additive measurement noise ϵ_i. The nonlinear dependence of z_i on $\boldsymbol{x}_i$ is approximated by a Bayesian neural network $f(\boldsymbol{x}_i, \boldsymbol{\phi})$ with weights and biases represented by latent (random) variables $\boldsymbol{\phi}$. The neural network is composed of L functions, $f = f^{(L)} \circ \cdots \circ f^{(1)}$, where $f^{(1)}$ to $f^{(L-1)}$ are called the hidden layers of f, and $f^{(L)}$ is the output layer (Goodfellow, Bengio, and Courville, 2016). A commonly used form of a hidden layer l is $f^{(l)}(\boldsymbol{x}) = \mathrm{ReLU}(W^{(l)}\boldsymbol{x} + \boldsymbol{b}^{(l)})$, where the rectified linear unit (ReLU) operator is given as $\mathrm{ReLU}(\boldsymbol{z})_i = \max\{z_i, 0\}$, $W^{(l)}$ is a weight matrix, and $\boldsymbol{b}^{(l)}$ is a vector of biases. For regression tasks the output layer is usually taken to be an affine mapping, $f^{(L)}(\boldsymbol{x}) = W^{(L)}\boldsymbol{x} + \boldsymbol{b}^{(L)}$. The layer weights and biases are collected in $\boldsymbol{\phi} = \{(W^{(l)}, \boldsymbol{b}^{(l)})\}_{l=1}^{L}$ to enable the compact notation $f(\boldsymbol{x}_i, \boldsymbol{\phi})$. With a slight abuse of this notation, an element ϕ_i of $\boldsymbol{\phi}$ represents a scalar weight or bias for $i \in \{1, \ldots, K_\phi\}$, where K_ϕ is the total number of weights and biases in the neural network. The distinguishing feature of a Bayesian neural network is that the weights and biases, $\boldsymbol{\phi}$, are modeled as random variables with a prior distribution $p(\boldsymbol{\phi})$.

We assume the noise to be normally distributed with standard deviation $g(z_i, \boldsymbol{\psi}) > 0$, and we consider different functions g of z_i and latent variables $\boldsymbol{\psi}$. We discuss the priors on the latent variables, $p(\boldsymbol{\phi})$ and $p(\boldsymbol{\psi})$, in the subsequent sections. The probabilistic model is illustrated graphically in Figure I.3.

Given $\boldsymbol{\phi}$, $\boldsymbol{\psi}$ and explanatory variables $\boldsymbol{x}$, the conditional flow rate $z = f(\boldsymbol{x}, \boldsymbol{\phi})$ and a measurement y is generated as

$$y \,|\, z, \boldsymbol{\psi} \sim \mathcal{N}(y \,|\, z, g(z, \boldsymbol{\psi})^2). \tag{I.2}$$

The flow rate measurement y is subject to epistemic (model) uncertainty in $f(\boldsymbol{x}, \boldsymbol{\phi})$ and aleatoric (measurement) uncertainty via $g(z, \boldsymbol{\psi})$. We differ between homoscedastic and heteroscedastic measurement noise. Heteroscedasticity is when the structure of the noise in a signal is dependent on the structure of the signal itself and is more difficult to capture (Woodward, Alsberg, and Kell, 1998). Homoscedasticity is the lack of heteroscedasticity.

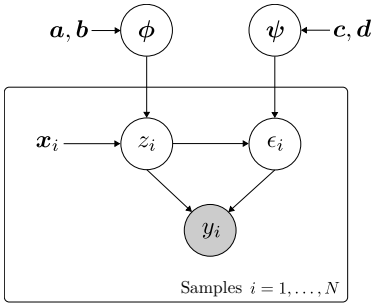


Figure I.3: A probabilistic graphical model for flow rates. Random variables are inscribed by a circle. A gray-filled circle means that the random variable is observed. The dependence $z_i \to \epsilon_i$ indicates that the noise is heteroscedastic, while the dependence $\boldsymbol{\psi} \to \epsilon_i$ indicates that the noise model is learned from data.

The flow model in (I.1) is a quite generic regression model, but it restricts the modeling of the measurement noise. The model allows the noise to be heteroscedastic, with the noise level being a function of the flow rate z, or homoscedastic for which the noise level is fixed. In the latter case, $g(z, \boldsymbol{\psi}) = \sigma_n$, where σ_n is a fixed noise level. If the noise level is unknown, it can be learned with the following homoscedastic noise model:

$$\begin{aligned} g(z_i, \boldsymbol{\psi}) &= \exp(\psi_1), \\ \psi_1 &\sim \mathcal{N}(c_1, d_1^2), \end{aligned} \tag{I.3}$$

where ψ_1 is a normally distributed latent variable and the noise level is log-normal. The exponential ensures that $g(z_i, \boldsymbol{\psi}) > 0$.

The homoscedastic noise model in (I.3) may be unrealistic for flow meters with a heteroscedastic noise profile. As described earlier, the uncertainty of the flow rate measurement is often given in relative terms. To model this property of the data, we augment (I.3) with a multiplicative term to get the following heteroscedastic noise model:

$$\begin{aligned} g(z_i, \boldsymbol{\psi}) &= \exp(\psi_2) \cdot |z_i| + \exp(\psi_1), \\ \psi_1 &\sim \mathcal{N}(c_1, d_1^2), \\ \psi_2 &\sim \mathcal{N}(c_2, d_2^2), \end{aligned} \tag{I.4}$$

where ψ_1 and ψ_2 are normally distributed latent variables[1]. Both $\exp(\psi_1)$ and $\exp(\psi_2)$ are log-normal, and are hence strictly positive. It follows from $|z| \geq 0$ that the noise standard deviation $g(z, \boldsymbol{\psi}) > 0$.

[1]We assume that we have one flow rate instrument for each well. Yet, several instruments may be handled by having separate noise models for each instrument.

I.4.1 Prior for the noise model, $p(\boldsymbol{\psi})$

The prior for the noise model is assumed to be a factorized normal

$$p(\boldsymbol{\psi}) = \prod_{i=1}^{K_\psi} \mathcal{N}(\psi_i \,|\, c_i, d_i^2), \tag{I.5}$$

where $K_\psi = 1$ for the homoscedastic noise model in (I.3) and $K_\psi = 2$ for the heteroscedastic noise model in (I.4).

The accuracy of an instrument measuring flow rate is commonly given as a mean absolute percentage error (MAPE) to a reference measurement. More precisely, the expected measurement error is specified as

$$\mathbf{E}_{y \,|\, z}\left[\frac{|y-z|}{|z|}\right] = E_r, \tag{I.6}$$

where y is the measurement, $z > 0$ is the reference measurement, and E_r is the MAPE, e.g. $E_r = 0.1$ for a MAPE of 10%. We wish to translate such statements to a prior $p(\boldsymbol{\psi})$.

Assuming a perfect reference measurement z, normal noise ϵ, and an additive noise model $y = z + \epsilon$, we obtain from (I.6) a noise standard deviation $g(z) = \sqrt{\pi/2}E_r|z|$. We recognize this as the first term in the heteroscedastic noise model (I.4). We derive prior parameters of $\psi_2 \sim \mathcal{N}(c_2, d_2^2)$ that correspond to a log-normal distribution $\exp(\psi_2)$ with mean $\sqrt{\pi/2}E_r$ by solving:

$$c_2 = \log(\sqrt{\pi/2}E_r) - d_2^2/2, \tag{I.7}$$

where we can adjust the variance d_2^2 to express our uncertainty in the value of E_r.

The specification of a relative measurement error E_r cannot be translated directly to a fixed noise level, as required by the homoscedastic noise model in (I.3). However, we can obtain a reasonable approximation by using the above procedure. If we set $z = \bar{z}$, where $\bar{z}$ is the mean production of a well, we can calculate prior parameters for ψ_1 as follows:

$$c_1 = \log(\sqrt{\pi/2}E_r\bar{z}) - d_1^2/2. \tag{I.8}$$

We express our uncertainty about the noise level by adjusting the variance d_1^2.

I.4.2 Prior for the neural network weights, $p(\boldsymbol{\phi})$

We encode our initial belief of the parameters $\boldsymbol{\phi}$ with a fully factorized normal prior

$$p(\boldsymbol{\phi}) = \prod_{i=1}^{K_\phi} \mathcal{N}(\phi_i \,|\, a_i, b_i^2), \tag{I.9}$$

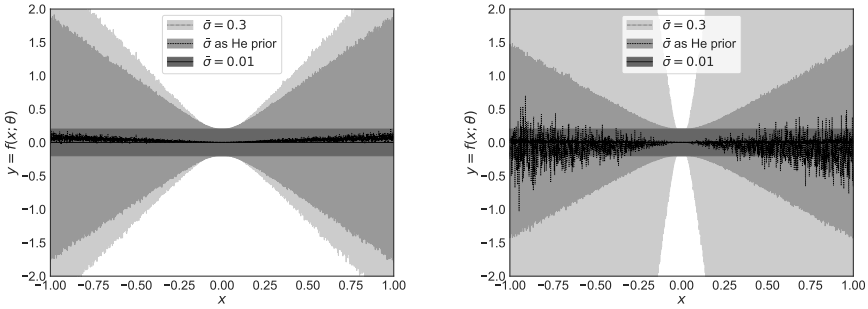
where K_ϕ is the number of weights and biases in the neural networks f. We assume a zero mean for the weights and biases, that is $a_i = 0$, as is common

practice for neural networks. One interpretation of the prior standard deviations is that they encode the (believed) frequencies of the underlying function, with low values of $\boldsymbol{b}$ inducing slow-varying (low frequency) functions, and high values inducing fast-varying (high frequency) functions (Gal, 2016). While this interpretation can give us some intuition about the effect of the prior, it is not sufficiently developed to guide the specification of a reasonable prior. We refrain from learning the prior from the data (as with empirical Bayes) and therefore treat $\boldsymbol{b}$ as hyperparameters to be prespecified.

For deep neural networks it is common practice to randomly sample the initial weights so that the output has a variance of one for a standard normal distributed input (Glorot and Bengio, 2010; He et al., 2015). For example, He-initialization (He et al., 2015) is often used for neural networks with ReLU activation functions. With He-initialization, the weights of layer l are drawn from the distribution $\mathcal{N}(0, \sigma_l^2)$ with $\sigma_l = \sqrt{2/n_l}$, where n_l is the number of layer inputs. The weights in the first hidden layer are initialized with $\sigma_l = \sqrt{1/n_l}$ since no ReLU activation is applied to the network's input. With layer biases set to zero, this initialization scheme yields a unit variance for the output.

The objective of weight initialization is similar to that of prior specification; a goal in both settings is to find a good initial model. In this work, we use the standard deviations $b_i = \sigma_l$ as a starting point for the prior specification (for weight i in layer l of a ReLU network). We call this the He-prior. The resulting standard deviations can then be increased (or decreased) if one believes that the underlying function amplifies (or diminishes) the input signal.

Figure I.4 shows the effect of $\boldsymbol{b}$ on the predictive uncertainty of a Bayesian neural network. With a common prior standard deviation (same for all weights), the output variance is sensitive to the network size (depth and width). This sensitivity complicates the prior specification, as illustrated for different network depths in the figure. The He-prior retains a unit output variance for different network sizes.



(a) Two hidden layers, each with 50 nodes (b) Five hidden layers, each with 50 nodes

Figure I.4: Prediction uncertainty (two sigma) for different priors $b_i = \bar{\sigma}$ on a neural network's weights. Two networks are trained on a dataset $\mathcal{D} = \{(0, y_i)\}_{i=1}^{100}$, where $y_i \sim \mathcal{N}(0, \sigma_n^2)$ and the noise level $\sigma_n = 0.1$ is known. The figure shows that the epistemic (model) uncertainty is explained away for $x = 0$ and increasing with the distance to $x = 0$. Away from the data, the increase in epistemic uncertainty depends on the prior variance and network depth.

I.4.3 A fully factorized normal prior on the latent variables

The prior of model (I.1) is a fully factorized normal distribution, $p(\boldsymbol{\phi})p(\boldsymbol{\psi})$. To simplify the notation in the rest of this paper we collect the latent variables in $\boldsymbol{\theta} = (\boldsymbol{\phi}, \boldsymbol{\psi}) \in \mathbb{R}^K$, where $K = K_\phi + K_\psi$. This allows us to state the prior on $\boldsymbol{\theta}$ as $p(\boldsymbol{\theta}) = p(\boldsymbol{\phi})p(\boldsymbol{\psi})$, where

$$p(\boldsymbol{\theta}) = \prod_{i=1}^{K} \mathcal{N}(\theta_i \,|\, \bar{\mu}_i, \bar{\sigma}_i^2), \tag{I.10}$$

with means $\bar{\boldsymbol{\mu}} = (\bar{\mu}_1, \ldots, \bar{\mu}_K) = (a_1, \ldots, a_{K_\phi}, c_1, \ldots, c_{K_\psi}) \in \mathbb{R}^K$ and standard deviations $\bar{\boldsymbol{\sigma}} = (\bar{\sigma}_1, \ldots, \bar{\sigma}_K) = (b_1, \ldots, b_{K_\phi}, d_1, \ldots, d_{K_\psi}) \in \mathbb{R}^K$. The total number of model parameters ($\bar{\boldsymbol{\mu}}$ and $\bar{\boldsymbol{\sigma}}$) is $2K$.

I.5 Methods

We wish to infer the latent variables $\boldsymbol{\theta}$ of the flow rate model in (I.1) from observed data. With Bayesian inference, the initial belief of $\boldsymbol{\theta}$, captured by the prior distribution $p(\boldsymbol{\theta})$ in (I.10), is updated to a posterior distribution $p(\boldsymbol{\theta} \,|\, \mathcal{D})$ after observing data $\mathcal{D}$. The update is performed according to Bayes' rule:

$$p(\boldsymbol{\theta} \,|\, \mathcal{D}) = \frac{p(\mathcal{D} \,|\, \boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathcal{D})}, \tag{I.11}$$

where $p(\mathcal{D})$ is the evidence and the likelihood is given by

$$p(\mathcal{D} \mid \boldsymbol{\theta}) = \prod_{i=1}^{N} p(y_i \mid \boldsymbol{x}_i, \boldsymbol{\theta}). \tag{I.12}$$

The log-likelihood of the model in (I.1) is shown in I.A.1.

From the posterior distribution, we can form the predictive posterior distribution

$$p(y^+ \mid \boldsymbol{x}^+, \mathcal{D}) = \int p(y^+ \mid \boldsymbol{x}^+, \boldsymbol{\theta}) p(\boldsymbol{\theta} \mid \mathcal{D}) d\boldsymbol{\theta} \tag{I.13}$$

to make a prediction y^+ for a new data point $\boldsymbol{x}^+$.

The posterior in (I.11) involves intractable integrals that prevents a direct application of Bayes' rule (Blei, Kucukelbir, and McAuliffe, 2017). In the following sections, we review two methods that circumvent this issue, namely maximum a posteriori (MAP) estimation and variational inference. With MAP estimation inference is simplified by considering only the mode of $p(\boldsymbol{\theta} \mid \mathcal{D})$, and with variational inference the posterior distribution is approximated. In the latter case, we can form an approximated predictive posterior distribution by replacing the posterior in (I.13) with its approximation. Statistics of this distribution, such as the mean and variance, can be estimated using Monte-Carlo sampling (Gal, 2016).

I.5.1 MAP estimation

With maximum a posteriori (MAP) estimation we attempt to compute:

$$\hat{\boldsymbol{\theta}}_{\text{MAP}} = \arg\max_{\boldsymbol{\theta}} \; p(\boldsymbol{\theta} \mid \mathcal{D}), \tag{I.14}$$

where $\hat{\boldsymbol{\theta}}_{\text{MAP}}$ is the mode of the posterior distribution in (I.11). For the model in (I.1) with a fixed and constant noise variance σ_n^2 and $\bar{\sigma}_i^2$ is the (prior) variance of θ_i, we have that

$$\begin{aligned} \hat{\boldsymbol{\theta}}_{\text{MAP}} &= \arg\max_{\boldsymbol{\theta}} \; \log p(\mathcal{D} \mid \boldsymbol{\theta}) + \log p(\boldsymbol{\theta}) \\ &= \arg\min_{\boldsymbol{\theta}} \; \frac{1}{2\sigma_n^2} \sum_{i=1}^{N} \left(y_i - f(\boldsymbol{x}_i, \boldsymbol{\theta})\right)^2 + \sum_{i=1}^{K} \frac{1}{2\bar{\sigma}_i^2} \theta_i^2, \end{aligned} \tag{I.15}$$

From (I.15), we see that MAP estimation is equivalent to maximum likelihood estimation with L^2-regularization (Hastie, Tibshirani, and Friedman, 2009).

While MAP estimation allows us to incorporate prior information about the model, it provides only a point estimate $\hat{\boldsymbol{\theta}}_{\text{MAP}}$ and will not capture the epistemic uncertainty of the model. To obtain a posterior distribution of $\boldsymbol{\theta}$ we consider the method of variational inference.

I.5.2 Variational inference

With variational inference, the posterior in (I.11) is approximated by solving an optimization problem, cf. Blei, Kucukelbir, and McAuliffe (2017). Consider a variational posterior density $q(\boldsymbol{\theta} \,|\, \boldsymbol{\lambda})$, parameterized by a real vector $\boldsymbol{\lambda}$. The objective of the optimization is to find a density $q^\star = q(\boldsymbol{\theta} \,|\, \boldsymbol{\lambda}^\star)$ that minimizes the Kullback-Leibler (KL) divergence to the exact posterior, i.e.

$$\boldsymbol{\lambda}^\star = \arg\min_{\boldsymbol{\lambda}} \, D_{\mathrm{KL}} \left(q(\boldsymbol{\theta} \,|\, \boldsymbol{\lambda}) \,\|\, p(\boldsymbol{\theta} \,|\, \mathcal{D}) \right). \tag{I.16}$$

A direct approach to solve (I.16) is not practical since it includes the intractable posterior. In practice, the KL divergence is instead minimized indirectly by maximizing the evidence lower bound (ELBO):

$$\begin{aligned} \mathcal{L}(\boldsymbol{\lambda}) &:= \log p(\mathcal{D}) - D_{\mathrm{KL}} \left(q(\boldsymbol{\theta} \,|\, \boldsymbol{\lambda}) \,\|\, p(\boldsymbol{\theta} \,|\, \mathcal{D}) \right) && \text{(I.17)} \\ &= \mathbf{E}_q \left[\log p(\mathcal{D} | \boldsymbol{\theta}) \right] - D_{\mathrm{KL}} \left(q(\boldsymbol{\theta} \,|\, \boldsymbol{\lambda}) \,\|\, p(\boldsymbol{\theta}) \right), && \text{(I.18)} \end{aligned}$$

where the expectation $\mathbf{E}_q \left[\cdot \right]$ is taken with respect to $q(\boldsymbol{\theta} \,|\, \boldsymbol{\lambda})$. From the ELBO loss in (I.18), we see that an optimal variational distribution maximizes the expected log-likelihood on the dataset, while obtaining similarity to the prior via the regularizing term $D_{\mathrm{KL}} \left(q(\boldsymbol{\theta} \,|\, \boldsymbol{\lambda}) \,\|\, p(\boldsymbol{\theta}) \right)$.

I.5.2.1 Stochastic gradient variational Bayes

Stochastic gradient variational Bayes (SGVB) or Bayes by backprop is an efficient method for gradient-based optimization of the ELBO loss in (I.18), cf. Blundell et al. (2015) and Kingma and Welling (2014).

Suppose that the variational posterior $q(\boldsymbol{\theta} \,|\, \boldsymbol{\lambda})$ is a mean-field (diagonal) normal distribution with mean $\boldsymbol{\mu}$ and standard deviation $\boldsymbol{\sigma}$. Let the variational parameters be $\boldsymbol{\lambda} = (\boldsymbol{\mu}, \boldsymbol{\rho})$ and compute $\boldsymbol{\sigma} = \log(1 + \exp(\boldsymbol{\rho}))$, where we use an elementwise softplus mapping to ensure that $\sigma_i > 0$.

The basic idea of SGVB is to reparameterize the latent variables to $\boldsymbol{\theta} = h(\boldsymbol{\zeta}, \boldsymbol{\lambda}) = \boldsymbol{\mu} + \log(1 + \exp(\boldsymbol{\rho})) \circ \boldsymbol{\zeta}$, where $\circ$ denotes pointwise multiplication and $\boldsymbol{\zeta} \sim \mathcal{N}(0, I)$. With this formulation, the stochasticity of $\boldsymbol{\theta}$ is described by a standard normal noise $\boldsymbol{\zeta}$ which is shifted by $\boldsymbol{\mu}$ and scaled by $\boldsymbol{\sigma}$. The reparameterization allows us to compute the gradient of the ELBO (I.18) as follows:

$$\begin{aligned} \nabla_{\boldsymbol{\lambda}} \mathcal{L}(\boldsymbol{\lambda}) &= \nabla_{\boldsymbol{\lambda}} \mathbf{E}_q \left[\log p(\mathcal{D} | \boldsymbol{\theta}) \right] - \nabla_{\boldsymbol{\lambda}} D_{\mathrm{KL}} \left(q(\boldsymbol{\theta} \,|\, \boldsymbol{\lambda}) \,\|\, p(\boldsymbol{\theta}) \right) \\ &= \mathbf{E}_\zeta \left[\nabla_{\boldsymbol{\theta}} \log p(\mathcal{D} \,|\, \boldsymbol{\theta}) \nabla_{\boldsymbol{\lambda}} h(\boldsymbol{\zeta}, \boldsymbol{\lambda}) \right] - \nabla_{\boldsymbol{\lambda}} D_{\mathrm{KL}} \left(q(\boldsymbol{\theta} \,|\, \boldsymbol{\lambda}) \,\|\, p(\boldsymbol{\theta}) \right) \end{aligned} \tag{I.19}$$

The expectation in (I.19) can be approximated by Monte-Carlo sampling the noise: $\boldsymbol{\zeta}_i \sim \mathcal{N}(0, I)$ for $i = 1, \ldots, M$. If we also approximate the likelihood by considering a mini-batch $\mathcal{B} \subset \mathcal{D}$ of size $B \leq N$, we obtain the unbiased SGVB estimator of the ELBO gradient:

$$\begin{aligned} \nabla_{\boldsymbol{\lambda}} \mathcal{L}(\boldsymbol{\lambda}) \simeq \nabla_{\boldsymbol{\lambda}} \hat{\mathcal{L}}(\boldsymbol{\lambda}) := \frac{N}{B} \frac{1}{M} \sum_{i=1}^{M} & \nabla_{\boldsymbol{\theta}} \log p(\mathcal{B} \,|\, \boldsymbol{\theta}) \nabla_{\boldsymbol{\lambda}} h(\boldsymbol{\zeta}_i, \boldsymbol{\lambda}) \\ & - \nabla_{\boldsymbol{\lambda}} D_{\mathrm{KL}} \left(q(\boldsymbol{\theta} \,|\, \boldsymbol{\lambda}) \,\|\, p(\boldsymbol{\theta}) \right). \end{aligned} \tag{I.20}$$

An advantage with the SGVB estimator in (I.20) is that we can utilize the gradient of the model $\nabla_{\boldsymbol{\theta}} \log p(\mathcal{B}|\boldsymbol{\theta})$ as computed by back-propagation. When both the variational posterior and prior are mean-field normals, as is the case for our model, $D_{\mathrm{KL}}\left(q(\boldsymbol{\theta} \,|\, \boldsymbol{\lambda}) \,\|\, p(\boldsymbol{\theta})\right)$ can be computed analytically as shown in I.A.2.

In Algorithm 1 we summarize the basic SGVB algorithm for mean-field normals and Monte-Carlo sample size of $M = 1$. We finally note that for variables representing weights of a neural network, we implement the local reparameterization trick in Kingma, Salimans, and Welling (2015) to reduce gradient variance and save computations (not shown in Algorithm 1).

Algorithm 1 Basic implementation of SGVB for mean-field normals ($M = 1$)

Require: data $\mathcal{D}$, model $p(\mathcal{D}, \boldsymbol{\theta}) = p(\mathcal{D} \,|\, \boldsymbol{\theta})p(\boldsymbol{\theta})$, parameters $\boldsymbol{\lambda} = (\boldsymbol{\mu}, \boldsymbol{\rho})$, learning rate α.

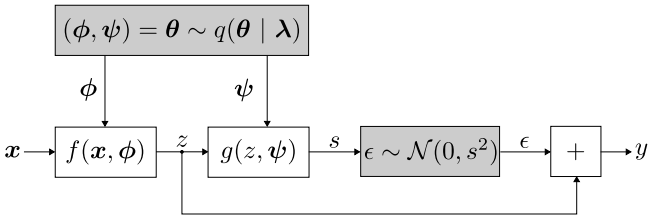
1: **repeat**
2: Sample mini-batch $\mathcal{B}$ from $\mathcal{D}$
3: Sample $\boldsymbol{\zeta} \sim \mathcal{N}(0, I)$
4: $\boldsymbol{\theta} \leftarrow \boldsymbol{\mu} + \log(1 + \exp(\boldsymbol{\rho})) \circ \boldsymbol{\zeta}$
5: Compute $\nabla_{\boldsymbol{\lambda}} \hat{\mathcal{L}}(\boldsymbol{\lambda})$ using (I.20)
6: $\boldsymbol{\lambda} \leftarrow \boldsymbol{\lambda} + \alpha \nabla_{\boldsymbol{\lambda}} \hat{\mathcal{L}}(\boldsymbol{\lambda})$
7: **until** no improvement in ELBO
8: **return** $\boldsymbol{\lambda}$

I.6 Case study

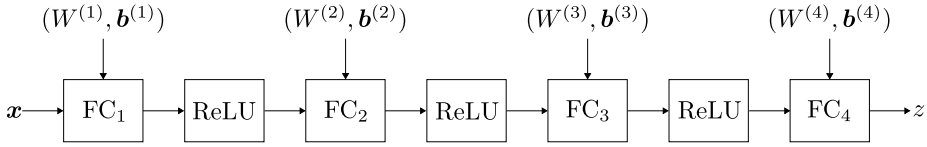
The goal of the case study was to investigate the predictive performance and generalization ability of the proposed VFM. The study was designed to test the predictive performance on historical data and on future data, which reflect the two main applications of a VFM. If the models generalize well, a similar performance across all wells for each model type should be expected on both historical and future data. To cast light on the data challenges in Section I.1, the results differentiate between wells with test separator and MPFM measurements, which have different measurement accuracy and frequency. The prediction uncertainty of the models was also analyzed and the effect of training set size on prediction performance was investigated.

The probabilistic flow rate models in Section I.4 were developed using the dataset described in Section I.3.1. The conditional mean flow rate, $f(\boldsymbol{x}, \boldsymbol{\phi})$, was modeled using a feed-forward neural network. Three different noise models were considered: a homoscedastic model with fixed noise standard deviation $g(z, \boldsymbol{\psi}) = \sigma_n = \text{const.}$, a homoscedastic model with learned noise standard deviation (I.3), and a heteroscedastic model with learned noise standard deviation (I.4). For each of the three model types and the 60 wells in the dataset, the neural network was trained using the SGVB method in Section I.5.2.1. These models will be referred to by the label VI-NN. For comparison, a neural network for each of the 60 wells was trained using the MAP estimation method in Section I.5.1.

For these models we considered the measurement noise to be homoscedastic with a fixed noise standard deviation (σ_n). We label these models as MAP-NN. The He-prior was used for the hidden layers to initialize and regularize the parameters, see Section I.4.2. For the noise models, we set the priors as described in Section I.4.1, differentiating between wells with MPFM and test separator measurements.



(a) Architecture of BNN



(b) Composition of $f(\boldsymbol{x}, \boldsymbol{\phi})$

Figure I.5: The architecture of the BNNs used in this study is illustrated in (a). Probabilistic computations are colored grey. Variables $\boldsymbol{\phi}$ and $\boldsymbol{\psi}$ are drawn from the approximate posterior and used to compute the conditional mean flow rate, $f(\boldsymbol{x}, \boldsymbol{\phi})$, and noise standard deviation, $g(z, \boldsymbol{\psi})$. The composition of $f(\boldsymbol{x}, \boldsymbol{\phi})$ with four layers (three hidden) and $\boldsymbol{\phi} = \{(W^{(l)}, \boldsymbol{b}^{(l)})\}_{l=1}^{4}$ is shown in (b). Fully connected blocks perform the operation $\mathrm{FC}_l(\boldsymbol{x}) = W^{(l)}\boldsymbol{x} + \boldsymbol{b}^{(l)}$.

A schematic representation of the Bayesian neural network is shown in Figure I.5. The network architecture was fixed to three hidden layers, each with 50 nodes to which we apply the ReLU activation function (Glorot, Bordes, and Bengio, 2011). Using practical recommendations in (Bengio, 2012), the network architecture may be large as long as regularization is used to prevent overfitting. The Adam optimizer (Kingma and Ba, 2015) with the learning rate set to 0.001 was used to train all networks. Early stopping with a validation dataset was used to determine an appropriate number of epochs to train the models to avoid overfitting (Goodfellow, Bengio, and Courville, 2016). The hyper-parameters were chosen by experimentation and using best practices. The models were implemented and trained using PyTorch (Paszke et al., 2019).

I.6.1 Prediction performance on historical data

To examine the predictive performance on historical data, a three months long period of contiguous data located in the middle of the dataset, when ordered chronologically, was set aside for testing. The rest of the data was used to train the models. During model development, a random sample of 20% of the training

data was used for model validation. The performance of each model type across the 60 wells was analyzed. Table I.1 shows the P_{10}, P_{25}, P_{50} (median), P_{75}, and P_{90} percentiles of the MAPE across all wells. Detailed results which differentiate between test separator and MPFM measurements are reported in I.B, Table I.4.

Table I.1: Prediction performance in terms of mean absolute percentage error on historical test data. The percentiles show the variation in performance among all wells.

Method and model	P_{10}	P_{25}	P_{50}	P_{75}	P_{90}
MAP-NN fixed homosc.	1.8	2.8	5.1	8.3	16.0
VI-NN fixed homosc.	1.4	2.6	4.8	8.5	12.8
VI-NN learned homosc.	1.3	2.4	5.3	8.4	13.3
VI-NN learned heterosc.	1.7	3.5	5.9	9.7	11.5

The results show that the four model types achieve similar performance to each other for the 75th and lower percentiles. The median MAPEs (P_{50}) lie in the range 4-6%. A comparison of the 90th percentile performance indicates that models trained by variational inference are more robust in terms of modeling difficult wells. Regardless of the model type used, there are large variations in the performance on different wells, as seen by comparing the 10th and 90th percentiles. The best performing model achieved an error of 0.3% for one of the wells. Yet, some models obtain an unsatisfactory large error. The overall worst-performing model (MAP-NN) achieved an error of 72.1% for one of the wells.

The cumulative performance of the four models is plotted in Figure I.6. The cumulative performance plot shows the percentage of test points that fall within a certain percent deviation from the actual measurements (Corneliussen et al., 2005). The figure shows that the models perform better on wells with MPFM measurements than on wells with test separator measurements. Again, similar performance of the four model types is observed.

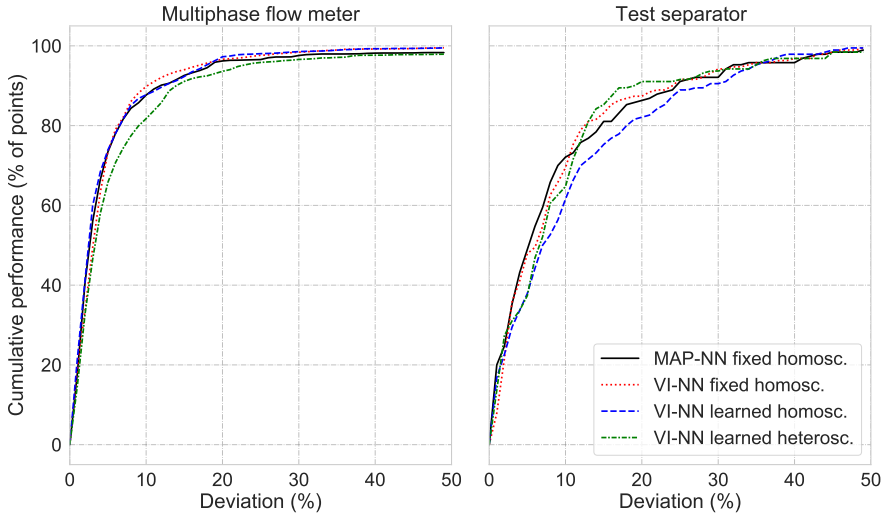


Figure I.6: Cumulative performance of the four models on historical test data. The cumulative performance is shown for wells with (left) MPFM and (right) test separator measurements.

I.6.2 Prediction performance on future data

The last three months of measurements were used to test the predictive performance on future data. The rest of the data was used to train the models. During model development, a random sample of 20% of the training data was used for model validation. Table I.2 shows the percentiles of the MAPE for the different models on all 60 wells. Detailed results which differentiate between MPFM and test separator measurements are given in I.B, Figure I.5.

Table I.2: Prediction performance in terms of mean absolute percentage error on future test data. The percentiles show the variation in performance among all wells.

Method and model	P_{10}	P_{25}	P_{50}	P_{75}	P_{90}
MAP-NN fixed homosc.	3.7	5.6	12.4	24.1	40.0
VI-NN fixed homosc.	4.0	5.6	9.6	18.2	29.3
VI-NN learned homosc.	4.0	6.0	8.9	22.5	32.5
VI-NN learned heterosc.	4.0	5.0	9.2	15.7	24.3

Similarly to the case with historical test data, the performance of the four model types is comparable for the 50th and lower percentiles. The median MAPEs (P_{50}) lie in the range 8-13%. For all model types, the 25% best-performing models achieved a MAPE of less than 6%. The best performing

model obtained a MAPE of 1.1% on one of the wells. This is in line with some of the best reported results in the literature; see Section I.2.1. Nevertheless, for each model type there is a large variation in performance among wells. The overall worst performing model achieved a MAPE of 48.7%.

Comparing the performance for either the 75th or 90th percentile again indicates that models trained by variational inference are more robust in terms of modeling difficult wells. In this regard, the heteroscedastic VI-NN performs particularly well compared to the other model types.

As seen from the cumulative performance plot in Figure I.7, the four model types have similar performance to each other. The exception is the heteroscedastic VI-NN, which outperforms the other model types for wells with test separator measurements. As seen in the case of historical test data, the models perform better on wells with MPFM measurements than on well with test separator measurements.

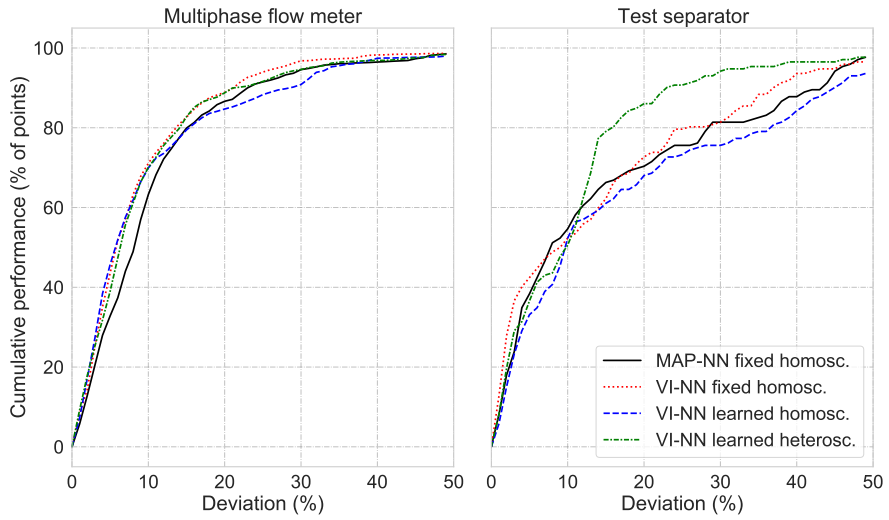


Figure I.7: Cumulative performance of the four models on future test data. The cumulative performance is shown for wells with (left) MPFM and (right) test separator measurements.

I.6.3 Comparison of performance on historical and future data

A comparison of the MAPEs on historical and future data is illustrated in Figure I.8. The plots differentiate wells with MPFM and test separator measurements. In general, the prediction error is larger on future test data than on historical test data. There is also a larger variance in the performance on future test data. This indicates that it is harder to make predictions on future data, than on

historical data. Further, observe that the errors are smaller for the wells with MPFM measurements than for the wells with test separator measurements in both the historical and future test data case.

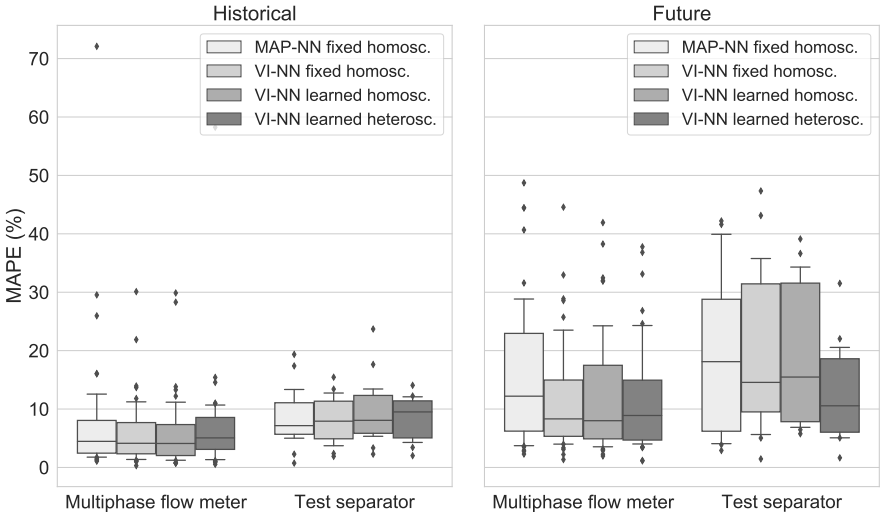
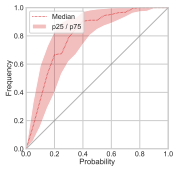


Figure I.8: Comparison of performance on historical and future data for the different models. The box plots differentiate between wells with multiphase flow meter and test separator measurements. The boxes show the P_{25}, P_{50} (median), and P_{75} percentiles. The whiskers show the P_{10} and P_{90} percentiles.

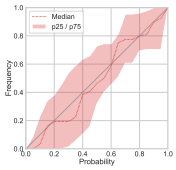
I.6.4 Uncertainty quantification and analysis

In contrary to the MAP-NN models, the VI-NN models quantify the uncertainty in their predictions. To study the quality of the prediction uncertainty, we generated a calibration plot for the three different noise models using the test datasets from Section I.6.1 and I.6.2; see Figure I.9. The plot shows the frequency of residuals lying within varying posterior intervals. For instance, for a perfectly calibrated model, 20% of the test points is expected to lie in the 20% posterior interval centered about the posterior mean. In other words, the calibration curve of a perfectly calibrated model will lie on the diagonal gray line illustrated in the figures. The calibration of a model may vary across wells. To visualize the variance in model calibration, we have illustrated the (point-wise) 25th and 75th percentiles of the calibration curves obtained across wells.

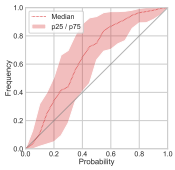
On historical data, the models trained on test separator measurements seem to be best calibrated. The models trained on MPFM measurements overestimate the uncertainty in their predictions. On future data, the results are reversed. The models trained on MPFM measurements are better calibrated and the



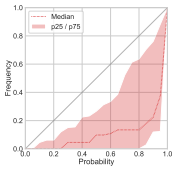
(a) MPFM, historical



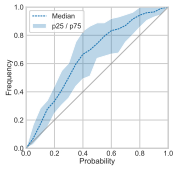
(b) Separator, historical



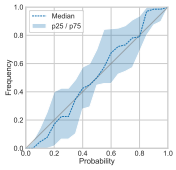
(c) MPFM, future



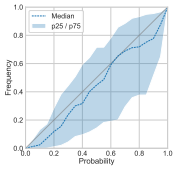
(d) Separator, future



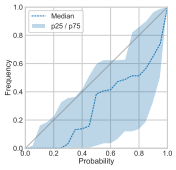
(e) MPFM, historical



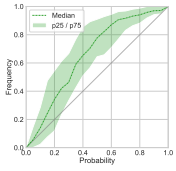
(f) Separator, historical



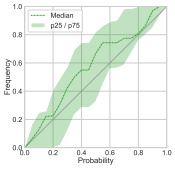
(g) MPFM, future



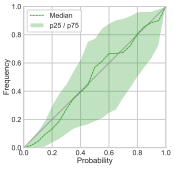
(h) Separator, future



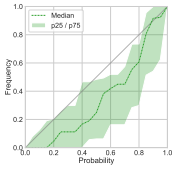
(i) MPFM, historical



(j) Separator, historical



(k) MPFM, future



(l) Separator, future

Figure I.9: Calibration plots for fixed homoscedastic noise (a-d), learned homoscedastic noise (e-h), and learned heteroscedastic noise (i-l). Wells are grouped by measurement device, multiphase flow meter or test separator, and the calibration on historical test data (Section I.6.1) and future test data (Section I.6.2) are shown. The median frequency is shown as a dashed line for each posterior interval (x-axis). The 25th and 75th percentiles (colored bands) show the variation in calibration across wells. A perfectly calibrated model would lie on the diagonal line $y = x$.

models trained on test separator measurements all underestimate the prediction uncertainty. Overall, the calibration improves when the noise model is learned. This is seen clearly when comparing the fixed homoscedastic noise to the learned heteroscedastic noise model. The results are summarized in Table I.3, which shows the coverage probabilities for the 95% posterior interval (using the pointwise median in the calibration plots).

Table I.3: Coverage probability (95%)

Case	Method and model	Test sep. (%)	MPFM (%)
Future prediction	VI-NN fixed homosc.	37.5	99.5
	VI-NN learned homosc.	81.0	87.7
	VI-NN learned heterosc.	92.3	90.0
Historical prediction	VI-NN fixed homosc.	92.4	100.0
	VI-NN learned homosc.	98.5	99.1
	VI-NN learned heterosc.	100.0	97.2

I.6.5 Effect of training set size on prediction performance

When analyzing the prediction performance of the four model types in Section I.6.1 and I.6.2, it was noticed that the prediction error tended to decrease as the training set size increased. This is illustrated in Figure I.10, which shows the MAPEs for the different models and corresponding regression lines with negative slopes. This tendency is generally expected of machine learning models. On the other hand, previous studies such as AL-Qutami, Ibrahim, and Ismail (2018), indicate that model performance does not necessarily improve when including data that is several years old. To closer inspect this effect, we compared models developed on successively larger training sets.

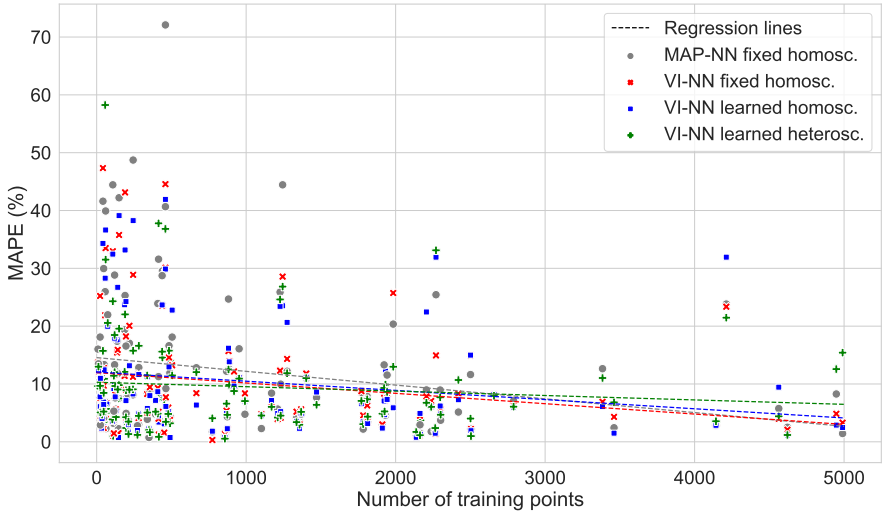


Figure I.10: The plot shows the mean absolute percentage error of the four models on historical and future test data for all wells. A regression line for each model shows the tendency of the error as the number of training points varies.

To allow for an interesting range of dataset sizes a subset of 21 wells with 1200 or more MPFM measurements was considered. In a number of trials, a well from the subset and an instant of time at which to split the dataset into a training and test set, were randomly picked. Keeping the test set fixed, a sequence of training sets of increasing size was generated. The training sets were extended backwards in time with data preceding the test data. The following training set sizes were considered: 150, 200, 300, ..., 1100, where the increment is 100 between 300 and 1100. A MAP-NN model was developed for each of these training sets, using early stopping and validating against the last 100 data points. The test set size was also set to 100 data points, spanning on average 90 days of production.

Denoting the test MAPE of the models by E_{150}, E_{200}, E_{300}, ..., E_{1100}, we computed relative MAPEs

$$R_k = \frac{E_k}{E_{150}}, \text{ for } k \in \{150, 200, 300, \ldots, 1100\}. \tag{I.21}$$

The relative errors indicate how the performance develops as the training set size increases, with a baseline at $R_{150} = 1$. The result of 400 trials is shown in Figure I.11.

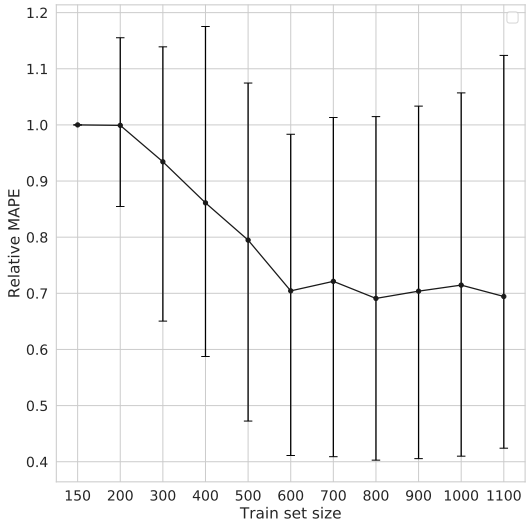


Figure I.11: Relative test errors of the MAP-NN model for increasing training set sizes. Shown are the medians and 50% intervals of 400 trials.

I.7 Discussion

In Section I.1 some of the challenges faced by data-driven VFMs were discussed. These were: (1) low data volume, (2) low data variety (3) poor measurement

quality, and (4) non-stationarity of the underlying process. Here we discuss the results in light of these challenges. All results are discussed in terms of MAPE values.

No widely used standard exists for VFM performance specification or requirements. Thus, the following performance requirements have been set by the authors to assess the commercial viability of a VFM: 1) predictive performance in terms of mean absolute percentage error on test data of 10% or less, and 2) robustness in terms of achieving the above predictive performance for at least 90% of wells. While these simple requirements lack a specification of the test data, we find them useful in the assessment of VFM performance. A VFM failing to meet these requirements would not be practical to use in industrial applications.

I.7.1 Performance on historical and future test data

First, we discuss the concern about the non-stationarity of the underlying process. This means the distribution of values seen during training is not necessarily the same as the distribution of values used for testing. The effect of this is best observed when comparing the performance on historical and future data, see Table I.1 and I.2 and Figure I.8. Looking at the upper and lower percentiles, we see the different models achieve performance in the range of 1-16% error on historical data and 3-40% error on future data. Since the strength of data-driven models lies with interpolation, rather than extrapolation, it is natural that the performance is worse on the future data case. Considering the VFM performance requirement of 10% MAPE for 90% of the wells, the performance is not acceptable for the historical or for the future data case. This indicates that the robustness of the models is inadequate for use in a commercial VFM. For real time applications, frequent model updates are likely required to achieve the VFM performance requirement. This raises the technical challenge of implementing a data-driven modeling approach.

The study on dataset size in Section I.6.5 further explores the development of data distributions and the effect older data has on future prediction errors. The result, seen in Figure I.11, indicates that additional data is only valuable up to a certain point, after which older data will no longer be useful when predicting future values. The point where this happens will naturally vary between wells. For the wells included here, this happens at 600 data points on average, for which the additional data is approximately 18 months or longer into the past. Looking at Figure I.10, we again see the trend that wells with more data perform better, but only up to a certain point. We remark that insufficient model capacity would have a similar effect on the performance. However, we find this to be unlikely in this case study due to the high capacity and low training errors of the neural networks used.

At this point we remark that, for two observations $D_1, D_2 \in \mathcal{D}$, we model conditional independence $(D_1 \perp\!\!\!\perp D_2 \mid \boldsymbol{\theta})$. While the observations result from preprocessing measurement data in a way that removes transients and decorrelates observations, we cannot guarantee independence due to the non-

stationary process. With dependant observations, the modeling assumption of conditional independence is not satisfied since the models lack temporal dependencies. This is also true for most, if not all published models for data-driven VFM. Models that include temporal dependencies may be better suited to learn from past data.

A second concern raised was related to small data regimes, both in terms of data volume and data variety. The results mentioned above also illustrate the effect of small data. Looking at Figure I.10, higher variance in performance is seen among wells with less than 700 data points. This is concerning because many of the wells, in particular those with test separator measurements as their primary source of data, have very few data points. Based on the median MAPE values in Figure I.8, also given in Table I.4 and I.5, models trained on MPFM data outperforms the models trained on test separator data. This indicates that data quantity may outweigh data quality in the small-data regime. The difference in performance is also evident in the cumulative performance plots, see Figure I.6 and I.7.

The wells that lie in the top quarter of performance achieved MAPE values comparable to the earlier works discussed in Section I.2.1. However, this performance seems difficult to achieve for the full set of wells. The difficulty in generalizing a single model architecture to a broad set of wells is troublesome for the potential commercialization of data-driven VFM.

I.7.2 Noise models

The last concern raised was poor data quality. In particular uncertainty in flow rate measurements, and potential gross errors in MPFM measurements.

The three different noise models perform similarly in terms of MAPE, on both historical and future data. The only exception being the learned heteroscedastic noise model, which performed better than the others on historical and future test data case when judged by the 90th percentile. This is believed to be because the heteroscedastic error term gives the objective function some added robustness towards large errors.

From the calibration plots in Figure I.9, we see that learning the noise model improves the calibration. The calibration curves for models trained on MPFM data generally lie above the curves for models trained on test separator data, both for historical and future predictions. This means that models trained on MPFM measurements are less confident in their predictions, even though they are trained on more data. It was suspected that models trained on MPFM data would reflect the increased uncertainty present in these measurements, but this is difficult to observe from the results. It is worth noting that the MPFM models are tested on MPFM data, so any systematic errors present in the MPFM measurements themselves will not be detected.

Because the models have potentially large prediction errors, especially for future data, it is desirable that the model can assess its performance. The coverage probabilities reported in Table I.3 give us some confidence in the

uncertainty estimates for the learned noise models, especially for the historical cases.

Neither the homoscedastic or heteroscedastic noise models in (I.3) and (I.4), respectively, can capture complex noise profiles that depend on the flow conditions $\boldsymbol{x}$. As most flow meters are specialized to accurately measure flow rates for certain compositions and flow regimes, this is a potential drawback of the models. We leave it to later works to address such limitations, but note that with few adjustments the flow model in (I.1) can accommodate heteroscedasticity of a rather general form.

I.7.3 Bayesian neural networks

As stated in Section I.1, setting the priors on the parameters in the model is not a trivial task. In several papers, the Kullback-Leibler divergence term of the ELBO loss in (I.18) is down-weighted to improve model performance due to poor priors (Wenzel et al., 2020). This remains a research question, however, in Section I.4.2 one way of approaching prior specification in BNNs is described. The difficulty of setting priors combined with small data sets may make it difficult to successfully train models of this complexity. Still, the results are reasonable in the historical data case, and the estimated uncertainty is still better than only relying on point estimates.

I.8 Concluding remarks

MAP estimation and VI for a probabilistic, data-driven VFM was presented and explored in a case study with 60 wells. The models achieve acceptable performance on future test data for approximately half of the studied wells. It is observed that models trained on historical data lack robustness in a changing environment. Frequent model updates are therefore likely required, which pose a technical challenge in terms of VFM maintenance.

Of the presented data challenges, the non-stationary data distribution is the most concerning. It means that models must have decent extrapolating properties if they are to be used in real-time applications. This is inherently challenging for data-driven approaches, and limits the performance of all the models considered in this paper. Of the models explored here, VI provided more robust predictions than MAP estimation on future test data.

The BNN approach is promising due to its ability to provide uncertainty estimates. Among these models, the heteroscedastic model had the best performance, indicating that a heteroscedastic model can be advantageous for flow rate measurements. However, it is challenging to obtain well-calibrated models due to the difficulty of setting meaningful priors on neural network weights, and the fact that priors play a significant role in small data regimes. As a result, the uncertainty estimates provided by the BNNs should be used with caution.

I.8.1 Recommendations for future research

We would suggest future research on data-driven VFM to focus on ways to overcome the challenges related to small data and non-stationary data distributions. Advances on these problems are likely required to improve the robustness and extrapolation capabilities of models to be used in real-time applications. We believe promising avenues of research to be: i) hybrid data-driven, physics-based models that allows for stronger priors; ii) data-driven architectures that enables learning from more data, for instance by sharing parameters between well models; iii) online learning to enable frequent model updates; and iv) modeling of temporal dependencies, for example using sequence models, to better capture time-varying boundary conditions.

Acknowledgements

This work was supported by Solution Seeker AS.

Appendix I.A Derivations

I.A.1 Log-likelihood of the flow rate model

The log-likelihood of the flow model in (I.1) with parameters $\boldsymbol{\theta} = (\boldsymbol{\phi}, \boldsymbol{\psi})$ on a dataset $\mathcal{D} = (X, \boldsymbol{y}) = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$ is given by

$$\begin{aligned}\log p(\boldsymbol{y} \mid X, \boldsymbol{\theta}) &= \sum_{i=1}^{N} \log p(y_i \mid \boldsymbol{x}_i, \boldsymbol{\theta}) \\ &= \sum_{i=1}^{N} \log \mathcal{N}(y_i \mid f(\boldsymbol{x}_i, \boldsymbol{\phi}), g(f(\boldsymbol{x}_i, \boldsymbol{\phi}), \boldsymbol{\psi})^2) \\ &= -\frac{N}{2}\log(2\pi) - \sum_{i=1}^{N} \log g(f(\boldsymbol{x}_i, \boldsymbol{\phi}), \boldsymbol{\psi}) - \frac{1}{2}\left(\frac{y_i - f(\boldsymbol{x}_i, \boldsymbol{\phi})}{g(f(\boldsymbol{x}_i, \boldsymbol{\phi}), \boldsymbol{\psi})}\right)^2 .\end{aligned} \tag{I.22}$$

With a homoscedastic noise model $g(z, \boldsymbol{\psi}) = \sigma_n = \text{const.}$, the log-likelihood simplifies to:

$$\log p(\boldsymbol{y} \mid X, \boldsymbol{\theta}) = -\frac{N}{2}\log(2\pi\sigma_n^2) - \frac{1}{2\sigma_n^2}\sum_{i=1}^{N}\left(y_i - f(\boldsymbol{x}_i, \boldsymbol{\phi})\right)^2 . \tag{I.23}$$

I.A.2 Kullback-Leibler divergence term, $D_{\mathsf{KL}}\left(q(\boldsymbol{\theta} \mid \boldsymbol{\lambda}) \,\|\, p(\boldsymbol{\theta})\right)$

Let the approximation $q(\boldsymbol{\theta} \mid \boldsymbol{\lambda})$ and prior $p(\boldsymbol{\theta})$ be mean-field normal distributions of the random variables $\boldsymbol{\theta} \in \mathbb{R}^K$. Assume that the approximation is parameterized with $\boldsymbol{\lambda} = (\boldsymbol{\mu}, \boldsymbol{\rho})$, where $\boldsymbol{\mu}$ is the mean and $\boldsymbol{\sigma} = \log(1 + \exp(\boldsymbol{\rho}))$ is

the standard deviation of q. Then, the Kullback-Leibler divergence is given as:

$$
\begin{aligned}
&D_{\mathrm{KL}}\left(q(\boldsymbol{\theta} \mid \boldsymbol{\lambda}) \,\|\, p(\boldsymbol{\theta})\right) = \mathbf{E}_q\left[\log q(\boldsymbol{\theta} \mid \boldsymbol{\lambda}) - \log p(\boldsymbol{\theta})\right] \\
&= \mathbf{E}_q\left[\sum_{i=1}^{K} \log q(\theta_i \mid \lambda_i) - \log p(\theta_i)\right] \\
&= \frac{1}{2}\mathbf{E}_q\left[\sum_{i=1}^{K} -\log(2\pi\sigma_i^2) - \left(\frac{\theta_i - \mu_i}{\sigma_i}\right)^2 + \log(2\pi\bar{\sigma}_i^2) + \left(\frac{\theta_i - \bar{\mu}_i}{\bar{\sigma}_i}\right)^2\right] \\
&= \frac{1}{2}\left[\sum_{i=1}^{K} -2\log\frac{\sigma_i}{\bar{\sigma}_i} - \frac{1}{\sigma_i^2}\underbrace{\mathbf{E}_{q_i}\left[(\theta_i - \mu_i)^2\right]}_{=\sigma_i^2} + \frac{1}{\bar{\sigma}_i^2}\mathbf{E}_{q_i}\left[(\theta_i - \bar{\mu}_i)^2\right]\right] \\
&= \frac{1}{2}\sum_{i=1}^{K}\left[-1 - 2\log\frac{\sigma_i}{\bar{\sigma}_i} + \frac{1}{\bar{\sigma}_i^2}\mathbf{E}_{q_i}\left[(\theta_i - \bar{\mu}_i)^2\right]\right] \\
&= \frac{1}{2}\sum_{i=1}^{K}\left[-1 - 2\log\frac{\sigma_i}{\bar{\sigma}_i} + \left(\frac{\mu_i - \bar{\mu}_i}{\bar{\sigma}_i}\right)^2 + \left(\frac{\sigma_i}{\bar{\sigma}_i}\right)^2\right]
\end{aligned}
\tag{I.24}
$$

Appendix I.B Results

Table I.4: Prediction performance on historical test data for each well group. Reported values are the P_{10}, P_{25}, P_{50}, P_{75}, and P_{90} percentiles for the statistics root mean square error (RMSE) and mean absolute percentage error (MAPE).

Well group	Method and model	RMSE	MAPE %
All	MAP-NN fixed homosc.	0.4, 0.7, 1.1, 1.7, 3.0	1.8, 2.8, 5.1, 8.3, 16.0
	VI-NN fixed homosc.	0.3, 0.5, 1.0, 2.1, 3.0	1.4, 2.6, 4.8, 8.5, 12.8
	VI-NN learned homosc.	0.3, 0.5, 1.0, 2.0, 3.0	1.3, 2.4, 5.3, 8.4, 13.3
	VI-NN learned heterosc.	0.4, 0.6, 1.2, 1.9, 3.0	1.7, 3.5, 5.9, 9.7, 11.5
Test sep.	MAP-NN fixed homosc.	0.4, 0.8, 1.5, 1.7, 3.0	3.1, 5.7, 7.2, 11.1, 16.2
	VI-NN fixed homosc.	0.5, 0.8, 1.6, 2.2, 4.3	2.8, 4.9, 7.9, 11.3, 13.2
	VI-NN learned homosc.	0.6, 1.1, 1.7, 2.1, 3.1	3.9, 5.8, 8.1, 12.3, 16.4
	VI-NN learned heterosc.	0.5, 1.0, 1.7, 2.1, 3.9	3.7, 5.1, 9.5, 11.4, 12.2
MPFM	MAP-NN fixed homosc.	0.3, 0.6, 1.0, 1.6, 2.8	1.8, 2.4, 4.5, 8.1, 14.3
	VI-NN fixed homosc.	0.3, 0.4, 1.0, 1.9, 2.9	1.3, 2.3, 4.1, 7.7, 11.5
	VI-NN learned homosc.	0.3, 0.4, 0.7, 1.6, 3.0	1.2, 2.0, 4.1, 7.3, 11.7
	VI-NN learned heterosc.	0.4, 0.5, 1.2, 1.5, 2.9	1.3, 3.1, 5.1, 8.6, 10.8

Table I.5: Prediction performance on future test data for each well group. Reported values are the P_{10}, P_{25}, P_{50}, P_{75}, and P_{90} percentiles for the statistics root mean square error (RMSE) and mean absolute percentage error (MAPE).

Well group	Method and model	RMSE	MAPE %
All	MAP-NN fixed homosc.	0.8, 1.2, 2.1, 4.0, 6.1	3.7, 5.6, 12.4, 24.1, 40.0
	VI-NN fixed homosc.	0.6, 1.1, 1.8, 3.5, 5.2	4.0, 5.6, 9.6, 18.2, 29.3
	VI-NN learned homosc.	0.7, 1.2, 1.9, 3.3, 5.5	4.0, 6.0, 8.9, 22.5, 32.5
	VI-NN learned heterosc.	0.6, 1.1, 1.7, 3.1, 4.5	4.0, 5.0, 9.2, 15.7, 24.3
Test sep.	MAP-NN fixed homosc.	0.8, 1.0, 1.6, 3.0, 6.7	3.9, 6.2, 18.1, 28.8, 41.1
	VI-NN fixed homosc.	0.3, 1.0, 2.1, 3.2, 8.0	5.2, 9.5, 14.6, 31.4, 40.9
	VI-NN learned homosc.	0.6, 1.3, 1.9, 3.6, 5.9	6.6, 7.8, 15.5, 31.6, 35.9
	VI-NN learned heterosc.	0.4, 1.2, 1.6, 2.3, 2.9	5.1, 6.0, 10.6, 18.6, 21.6
MPFM	MAP-NN fixed homosc.	0.9, 1.2, 2.4, 4.2, 5.7	3.7, 6.2, 12.2, 23.0, 30.2
	VI-NN fixed homosc.	0.8, 1.3, 1.8, 3.5, 4.6	4.0, 5.3, 8.3, 15.0, 24.6
	VI-NN learned homosc.	0.7, 1.1, 1.9, 3.1, 5.2	3.4, 4.9, 8.0, 17.5, 28.1
	VI-NN learned heterosc.	0.7, 1.0, 1.8, 3.3, 4.6	3.8, 4.7, 8.9, 14.9, 24.5

References

Ahmadi, M. A. et al. (2013). "Evolving artificial neural network and imperialist competitive algorithm for prediction oil flow rate of the reservoir". In: *Applied Soft Computing* vol. 13 (2), pp. 1085–1098.

Amin, A. (2015). "Evaluation of Commercially Available Virtual Flow Meters (VFMs)". In: *Proceedings of the Annual Offshore Technology Conference*, pp. 1293–1318.

Balaji, K. et al. (2018). "Status of data-driven methods and their application in oil and gas industry". In: *EAGE Conference and exhibition, Society of Petroleum Engineers*, pp. 1–20.

Bassamzadeh, N. and Ghanem, R. (2018). "Probabilistic data-driven prediction of wellbore signatures in high-dimensional data using Bayesian networks". In: *Society of Petroleum Engineers*.

Bello, O., Ade-Jacob, S., and Yuan, K. (2014). "Development of hybrid intelligent system for virtual flow metering in production wells". In: *SPE intelligent energy conference & exhibition*. Society of Petroleum Engineers.

Bengio, Y. (2012). "Practical recommendations for gradient-based training of deep architectures". In: *Neural networks: Tricks of the trade*. Springer, pp. 437–478.

Berneti, S. M. and Shahbazian, M. (2011). "An Imperialist Competitive Algorithm - Artificial Neural Network Method to Predict Oil Flow Rate of the Wells". In: *International Journal of Computer Applications* vol. 26 (10), pp. 47–50.

Bieker, H. P., Slupphaug, O., and Johansen, T. A. (2007). "Well management under uncertain gas or water oil ratios". In: *SPE Digital Energy Conference and Exhibition*. Society of Petroleum Engineers.

Bikmukhametov, T. and Jäschke, J. (2019). "Oil Production Monitoring Using Gradient Boosting Machine Learning Algorithm". In: *12th IFAC Symposium on Dynamics and Control of Process Systems, including Biosystems*. Vol. 52. IFAC-PapersOnLine, pp. 514–519.

— (2020). "First Principles and Machine Learning Virtual Flow Metering: A Literature Review". In: *Journal of Petroleum Science and Engineering* vol. 184.

Blei, D. M., Kucukelbir, A., and McAuliffe, J. D. (2017). "Variational Inference: A Review for Statisticians". In: *Journal of the American Statistical Association* vol. 112, no. 518, pp. 859–877.

Blundell, C. et al. (2015). "Weight Uncertainty in Neural Networks". In: *32nd International Conference on Machine Learning*. Vol. 37, pp. 1613–1622.

Câmara, M. M. et al. (2017). "Numerical Aspects of Data Reconciliation in Industrial Applications". In: *Processes* vol. 5 (4).

Corneliussen, S. et al. (2005). *Handbook of multiphase flow metering*. The Norwegian Society for Oil and Gas Measurements.

Fonseca, J. R., Gonçalves, M., and Azevedo, L. (2009). "Consideration of uncertainty in simulation and flow. In Portugese: Consideração de incerteza nas simulações de elevação e Escoamento". In: *An. do IV Semin. Elev. Artif. e Escoamento*.

Foss, B., Knudsen, B. R., and Grimstad, B. (2018). "Petroleum production optimization – A static or dynamic problem?" In: *Computers & Chemical Engineering* vol. 114, pp. 245–253.

Gal, Y. (2016). "Uncertainty in Deep Learning". PhD Thesis. University of Cambridge, p. 174.

Ghahramani, Z. (2015). "Probabilistic machine learning and artificial intelligence". In: *Nature* vol. 521 (7553), pp. 452–459.

Glorot, X. and Bengio, Y. (2010). "Understanding the difficulty of training deep feedforward neural networks". In: *Proceedings of the 13th international conference on artificial intelligence and statistics*. Vol. 9, pp. 249–256.

Glorot, X., Bordes, A., and Bengio, Y. (2011). "Deep Sparse Rectifier Neural Networks". In: *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*. Vol. 15, pp. 315–323.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press.

Grimstad, B. et al. (2016). "A Simple Data-Driven Approach to Production Estimation and Optimization". In: *SPE Intelligent Energy International Conference and Exhibition*.

Hasanvand, M. Z. and Berneti, S. (2015). "Predicting Oil Flow Rate due to Multiphase Flow Meter by Using an Artificial Neural Network". In: *Energy Sources, Part A: Recovery, Utilization, and Environmental Effects* vol. 37 (8), pp. 840–845.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning*. New York, USA: Springer.

He, K. et al. (2015). "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification". In: *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034.

Humphrey, G. B. et al. (2016). “A hybrid approach to monthly streamflow forecasting: Integrating hydrological model outputs into a Bayesian artificial neural network”. In: *Journal of Hydrology* vol. 540, pp. 623–640.

Hüllermeier, E. and Waegeman, W. (Mar. 2021). “Aleatoric and epistemic uncertainty in machine learning: an introduction to concepts and methods”. In: *Machine Learning* vol. 110, no. 3, pp. 457–506.

Jansen, J.-D. (2015). *Nodal Analysis of Oil and Gas Wells - Theory and Numerical Implementation*. TU Delft, The Netherlands: Delft University of Technology.

Kingma, D. P. and Welling, M. (2014). “Auto-Encoding Variational Bayes”. In: *2nd International Conference on Learning Representations (ICLR)*.

Kingma, D. P. and Ba, J. L. (2015). “Adam: A method for stochastic optimization”. In: *3rd International Conference on Learning Representations (ICLR)*, pp. 1–15.

Kingma, D. P., Salimans, T., and Welling, M. (2015). “Variational dropout and the local reparameterization trick”. In: *28th International Conference on Neural Information Processing Systems*. Vol. 2, pp. 2575–2583.

Krejbjerg, K. et al. (2019). *Conversion of multiphase meter flowrates*. The Norwegian Society for Oil and Gas Measurements (NFOGM).

Liu, Y. et al. (2012). “Data-driven based model for flow prediction of steam system in steel industry”. In: *Information Sciences* vol. 193, pp. 104–114.

Lorentzen, R. J., Stordal, A. S., Luo, X., et al. (2016). “Estimation of Production Rates by Use of Transient Well-Flow Modeling and the Auxiliary Particle Filter: Full-Scale Applications”. In: *SPE Production and Operations* vol. 31(2), pp. 163–175.

Lorentzen, R. J., Stordal, A. S., Nævdal, G., et al. (2014). “Estimation of Production Rates With Transient Well-Flow Modeling and the Auxiliary Particle Filter”. In: *Society of Petroleum Engineers* vol. 19(1), pp. 172–180.

Luo, X. et al. (2014). “Toward an enhanced Bayesian estimation framework for multiphase flow soft-sensing”. In: *Inverse Problems* vol. 30.

Marshall, C. and Thomas, A. (Sept. 2015). “Maximising Economic Recovery - a review of well test procedures in the North Sea”. In: *Offshore Europe Conference and Exhibition, Society of Petroleum Engineers*.

Mathilde, H., Bjarne, G., and Imsland, L. (2020). “Developing a Hybrid Data-Driven, Mechanistic Virtual Flow Meter - a Case Study”. In: *IFAC-PapersOnLine* vol. 53, no. 2. 21th IFAC World Congress, pp. 11692–11697.

Mishra, S. and Datta-Gupta, A. (2018). *Applied Statistical Modeling and Data Analytics - A Practical Guide for the Petroleum Geosciences*. Elsevier.

Monteiro, D. D., Chaves, G. S., et al. (2017). “Uncertainty analysis for production forecast in oil wells”. In: *SPE Latin American And Caribbean Petroleum Engineering Conference*.

Monteiro, D. D., Duque, M. M., et al. (2020). “Using Data analytics to quantify the impact of production test uncertainty on oil flow rate forecast”. In: *IFP Energies Nouvelles* vol. 75 (7), pp. 1–15.

Ovadia, Y. et al. (2019). “Can You Trust Your Model’s Uncertainty? Evaluating Predictive Uncertainty Under Dataset Shift”. In: *33rd Conference on Neural Information Processing Systems*. Vol. 32.

Paszke, A. et al. (2019). "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems* vol. 32, pp. 8026–8037.

Polson, N. G. and Sokolov, V. (2017). "Deep learning: A Bayesian perspective". In: *Bayesian Analysis* vol. 12, no. 4, pp. 1275–1304.

AL-Qutami, T. A., Ibrahim, R., and Ismail, I. (2018). "Virtual multiphase flow metering using diverse neural network ensemble and adaptive simulated annealing". In: *Expert Systems With Applications* vol. 93, pp. 72–85.

AL-Qutami, T. A., Ibrahim, R., Ismail, I., and Ishak, M. A. (2017a). "Development of soft sensor to estimate multiphase flow rates using neural networks and early stopping". In: *International Journal on Smart Sensing and Intelligent Systems*. Vol. 10, pp. 199–222.

— (2017b). "Hybrid neural network and regression tree ensemble pruned by simulated annealing for virtual flow metering application". In: *IEEE International Conference on Signal and Image Processing Applications (ICSIPA)*, pp. 304–309.

— (2017c). "Radial basis function network to predict gas flow rate in multiphase flow." In: *Proceedings of the 9th International Conference on Machine Learning and Computing*, pp. 141–146.

Snoek, J., Larochelle, H., and Adams, R. P. (2012). "Practical Bayesian Optimization of Machine Learning Algorithms". In: *Proceedings of the 25th International Conference on Neural Information Processing Systems*. Vol. 25, pp. 2951–2959.

Solle, D. et al. (2017). "Between the Poles of Data-Driven and Mechanistic Modeling for Process Operation". In: *Chemie Ingenieur Technik* vol. 89, no. 5, pp. 542–561.

Toskey, E. (2012). "Improvements to Deepwater Subsea Measurements RPSEA Program: Evaluation of Flow Modeling". In: *Proceedings of the Annual Offshore Technology Conference*, pp. 1–18.

Wenzel, F. et al. (2020). "How Good is the Bayes Posterior in Deep Neural Networks Really?" In: *Proceedings of the 37th International Conference on Machine Learning*. Vol. 119, pp. 10248–10259.

Woodward, A. M., Alsberg, B. K., and Kell, D. B. (1998). "The effect of heteroscedastic noise on the chemometric modelling of frequency domain data". In: *Chemometrics and Intelligent Laboratory Systems* vol. 40, pp. 101–107.

Xu, L. et al. (2011). "Wet gas metering using a revised venturi meter and soft-computing approximation techniques". In: *IEEE transactions on instrumentation and measurement* vol. 60 (3), pp. 947–956.

Zangl, G., Hermann, R., and Christian, S. (2017). "Comparison of methods for stochastic multiphase flow rate estimation". In: *SPE Annual Technical Conference and Exhibition* vol. 12.

Paper II

Multi-task learning for virtual flow metering

Anders T. Sandnes, Bjarne Grimstad, Odd Kolbjørnsen

Published in *Knowledge-Based Systems*, Volume 232, 28 November 2021, 107458 DOI: https://doi.org/10.1016/j.knosys.2021.107458

II

Abstract

Virtual flow metering (VFM) is a cost-effective and non-intrusive technology for inferring multiphase flow rates in petroleum assets. Inferences about flow rates are fundamental to decision support systems that operators extensively rely on. Data-driven VFM, where mechanistic models are replaced with machine learning models, has recently gained attention due to its promise of lower maintenance costs. While excellent performances in small sample studies have been reported in the literature, there is still considerable doubt about the robustness of data-driven VFM. In this paper, we propose a new multi-task learning (MTL) architecture for data-driven VFM. Our method differs from previous methods in that it enables learning across oil and gas wells. We study the method by modeling 55 wells from four petroleum assets and compare the results with two single-task baseline models. Our findings show that MTL improves robustness over single task methods, without sacrificing performance. MTL yields a 25-50% error reduction on average for the assets where single-task architectures are struggling.

II.1 Introduction

Knowledge of gas, oil, and water flow rates in a petroleum asset is highly valuable in operations and production planning but challenging to obtain (Hansen, Pedersen, and Durdevic, 2019). There is a large economic incentive for operators to maintain high production rates and avoid operational problems. Flow rates from individual wells support many important operational decisions, such as production optimization (Foss, Knudsen, and Grimstad, 2018), reservoir management (Kanshio, 2020), and flow assurance (Jamaluddin and Kabir, 2012).

Most assets consist of a set of wells that produce to a shared processing facility, as illustrated in Figure II.1. The joint flow from all wells is continuously measured after being physically separated into its main phases, gas, oil, and water. These can be accurately measured by single phase flow sensors. Flow

rates from individual wells are conventionally measured by routing the flow to a dedicated test separator. The resulting observations, known as *well tests*, are of high quality (Corneliussen et al., 2005). However, the frequency of well tests is low since the test separator accommodates one well at a time and requires several hours to measure the flow. It is therefore desirable to measure well flow rates before separation.

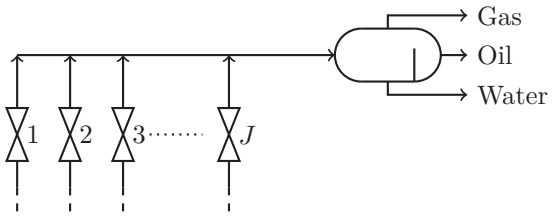


Figure II.1: Asset with J wells sharing a single separator.

There are two main strategies for measuring multiphase flow, multiphase flow meters (MPFM) and virtual flow meters (VFM) (Bikmukhametov and Jäschke, 2020). MPFMs are complex and expensive measurement devices physically installed in the well. VFM is a soft sensing technology that makes inferences about flow rates from existing sensor data and mathematical models implemented in software. VFM is often seen as complementary to MPFMs. Multiphase flow measurements have higher uncertainty than single phase measurements. Single phase measurements have errors around 0.25% for oil and 1% for gas rates (Thorn, Johansen, and Hjertaker, 2012). The quality and availability of water measurements are more varying. A calibrated MPFM is expected to have approximately 5% error for all phases(Thorn, Johansen, and Hjertaker, 2012). However, they are specialized to certain operating conditions and must be re-calibrated as conditions change(Corneliussen et al., 2005).

VFMs can be categorized based on their use of mechanistic or data-driven models (Bikmukhametov and Jäschke, 2020). A *mechanistic VFM* is derived from first principles and utilizes empirical correlations sparingly. A *data-driven VFM* is based on a machine learning method that fits a generic mathematical model to data. The generic models do not offer a physical interpretation of the parameters, as opposed to mechanistic models where parameters are related to physical properties. Most VFM solutions today are based on mechanistic models implemented in multiphase flow simulators (Amin, 2015). There are few, if any, commercially available data-driven VFM solutions. However, there has been an increasing interest in their development (Bikmukhametov and Jäschke, 2020), which is likely motivated by several factors. First, both instrumentation and data availability have improved. Second, the tooling for machine learning has improved considerably and the number of practitioners has increased. Third, oil and gas profit margins have decreased, leading to a search for more cost-efficient solutions. Data-driven VFM is attractive in terms of cost efficiency due to the promise of low maintenance requirements and high scalability. Data-driven

VFMs are expected to be easier to develop and maintain since they only require flow rate observations to be calibrated (AL-Qutami, Ibrahim, Ismail, and Ishak, 2018). This is in contrast to mechanistic models, which can be challenging to maintain due to high model complexity (Stenhouse, 2008). Calibration demands flow rate observations and experiment data, such as fluid samples, to attain physically meaningful parameters values. Furthermore, calibration often requires personnel with asset experience and expert knowledge of multiphase flow physics and the VFM software.

A diverse set of methods, models, and experiment setups for data-driven VFM have been presented in the literature. The recent survey in (Bikmukhametov and Jäschke, 2020) tabulates a selection of the proposed solutions, where architectures based on neural networks are the most frequent. Neural networks have been researched extensively and have been successfully applied in other domains, such as image analysis (Hong, Yu, Wan, et al., 2015; Yu et al., 2019), medicine (Hannun et al., 2019), and natural language processing (Vaswani et al., 2017), which motivates its popularity in VFM applications. Representative works on neural network based VFMs include (AL-Qutami, Ibrahim, Ismail, and Ishak, 2017; AL-Qutami, Ibrahim, Ismail, and Ishak, 2018), which report 2.2–4.2% errors and 2.4–4.7% errors respectively. A hybrid solution of neural networks and regression trees is presented in (AL-Qutami, Ibrahim, and Ismail, 2017), reporting errors in the range of 1.5–6.5%. Gradient boosted trees are explored as an alternative to neural networks in (Bikmukhametov and Jäschke, 2019), achieving errors of 2–6% in different scenarios. In all approaches, the VFM model is trained on data from a single well.

Several of the proposed solutions rival the expected performance of conventional MPFMs, but commercially viable alternatives have yet to emerge. The recent study in (Grimstad, Hotvedt, et al., 2021) applied Bayesian neural networks to data-driven VFM. The authors questioned if a robust data-driven method can be obtained by individually modeling wells from historical observations. Several challenges facing any data-driven VFM were highlighted. To reiterate, there are usually few data points for each individual well, making it difficult to identify complex models. Additionally, the underlying process is non-stationary, which makes past data less relevant for future predictions. Finally, the operational practices on most assets may result in low data variety and create highly correlated explanatory variables.

Challenges related to insufficient data are common in machine learning. A solution is to utilize data collected from other related problems (Lu et al., 2015; Y. Zhang and Yang, 2021). There are several ways such data could be combined. One approach is Multi-Task Learning (MTL), where models for all problems are jointly optimized (Goodfellow, Bengio, and Courville, 2016; Y. Zhang and Yang, 2021). In MTL, the problem is given as a set of tasks, $\{\mathcal{T}_1, \ldots, \mathcal{T}_J\}$, where each task $\mathcal{T}_j$ has a set of observations $(y_{ij}, x_{ij}), i = 1, \ldots, N_j$. MTL attempts to jointly learn models for each task, utilizing the knowledge from other tasks to improve performance. Tasks are assumed to share some common structure that enables the transfer of knowledge. Several mechanisms have been suggested to facilitate knowledge sharing. One approach is to have a set of parameters shared

between the task models.

Methods that combine MTL and deep learning have been successfully applied to several domains, e.g., image analysis (Hong, Yu, J. Zhang, et al., 2019), natural language processing (Sigtia et al., 2020), and speech processing (Majumder et al., 2019). It has also been applied to problems in the energy sector, such as solar and wind power (Dorado-Moreno et al., 2020; Wu, Li, and Xia, 2021). Multi-task neural networks have been presented in a wide range of complexities, from simple feed forward networks (Caruana, 1997) to more complex recent architectures that utilize both recurrent and convolutional network components (Jin et al., 2020). Some architectures, such as Cross-stitch networks, use a deep neural network for each task and are not designed to scale to numerous tasks (Misra et al., 2016). On the opposite side, context-sensitive networks use a task encoding as input to a network with all parameters being shared(Silver, Poirier, and Currie, 2008). A related approach is context adaptation, in which context parameters are learned and used as inputs to a shared neural network(Zintgraf et al., 2019). While much work has centered around neural networks, other learners such as support vector machines (Mei and Xu, 2020a; Mei and Xu, 2020b), and Gaussian process regression (Zhou et al., 2021) have also been successfully explored.

Even though knowledge sharing has been successful in many cases, it is not guaranteed that all tasks will benefit from each other (Standley et al., 2020). Negative transfer refers to the phenomenon where the performance of one task is reduced when another task is introduced. Deciding which task that should be learning together, and how to best avoid negative transfer, is still an open problem.

We present a multi-task learning based data-driven VFM. Our key insight is that knowledge can be shared among wells in a data-driven model, similarly to how knowledge is encoded and reused in mechanistic models. In the context of VFMs, we consider modeling the flow rate from one well as a learning task. Task domains have different data distributions (domain shift) and the tasks must learn different discriminative models. Our MTL architecture, which resembles that of (Zintgraf et al., 2019), is specialized for data-driven VFM, for which there is a large number of tasks with few observations. It utilizes well-specific parameters to adapt the domains and tasks. Domain adaptation is performed by learning domain-specific feature mappings, which transform input features to abstracted *domain features*. Task adaptation is enabled by learning *task-specific parameters*. The domain features and task parameters are fed to a shared discriminator, to predict flow rates. Because our architecture efficiently scales to many tasks, all wells can be modeled simultaneously.

The framing of data-driven VFM as an MTL problem enables us to learn from more data. While previous methods are limited to small datasets with observations from individual wells, our method scales learning to datasets with observations from any number of wells. To test the proposed method, we perform a study of 55 wells from four assets.

II.2 Problem description

The system of interest is the well choke valve. Choke valves are adjustable restrictions that are used to control the flow rate from the well. We only consider measurements that are commonly available for oil and gas wells. These are the pressure (p_1) and temperature (T) upstream the choke, the pressure downstream the choke (p_2), and the choke opening (u). In addition, flow rates (q) are measured by a separator (well testing) or an MPFM. A single choke valve is illustrated in Figure II.2.

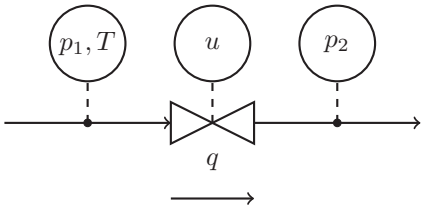


Figure II.2: Choke valve with instrumentation.

Flow rates are represented as a vector $q^T = \begin{bmatrix} q_G, q_O, q_W \end{bmatrix}$ of gas, oil, and water rate. The rates are customarily given in volumetric flow pr. day, at standard conditions (AIME and Society of Petroleum Engineers, 1984). However, due to the large magnitude of volumetric gas rates, q_G is scaled down by a factor of 1000 to represent liquid equivalents. We denote the total flow rate by $Q = q_G + q_O + q_W$, and the flow composition fractions by $\phi = q/Q$. Flow composition ϕ is dependent on reservoir conditions and is slowly time varying. It can be estimated or assumed fixed between well tests. Here we consider ϕ to be known.

We consider the problem of modelling Q given u, p_1, p_2, T, and ϕ. The gas, oil, and water flow rates are then found as $q = Q\phi$.

II.2.1 Insights from mechanistic modelling

The system in question poses some challenges that are best explored by a simple mechanistic example. For single phase flow, an analytic model

$$Q = AC\sqrt{\frac{p_1 - p_2}{\rho}}, \tag{II.1}$$

can be derived from the Bernoulli equation (White, 2008). Here, A is the choke opening area, C is a choke specific flow factor, and ρ is the fluid density. Equation II.1 is the result of generic assumptions and simplifications, and appears in multiple domains. Multiphase extensions to Equation II.1 are usually domain specific. Multiplier models are one class of such extensions for oil and gas flows. They introduce additional factors to Equation II.1 to correct errors in the pressure drop calculation due to multiphase flow. Additionally, the single

phase density is replaced by a mixture density. There are several variations of multiplier and density computations, some of which are explored in (Schüller, Solbakken, and Selmer-Olsen, 2003). These computations often rely on flow composition and fluid properties such as single phase densities.

Equation II.1 contains choke area A as one of the observed variables, and flow factor C as a given constant. However, these quantities are rarely measured directly. In the measurement setup considered here, the choke position is given in percent of full travel. Choke position is not directly comparable between wells, because they have different choke valve designs. It is common to describe choke valves by a CV curve (Grace and Frawley, 2011). The CV curve is a mapping between a choke opening and a flow factor, which captures the effect opening area and geometry have on an idealized flow rate. All mechanistic simulators include CV curves or similar mappings. Data-driven approaches often circumvent this by modeling directly on u, which means the mapping is implicit within a black box model.

To utilize a shared discriminator, it is necessary to adapt the observed values to universally comparable quantities and to capture the unique aspects of each well, such as fluid properties and choke geometries.

II.3 Data

Our data is a set of observations $(Q_{ij}, x_{ij}), i = 1, \dots N_j$, $j = 1, \dots, J$. Each data point is one observation from one well, indexed as data point i from well j. The total flow rate Q_{ij} is a scalar. Variables x_{ij} is a vector,

$$x_{ij}^{\top} = \left[u_{ij}, p_{ij,1}, p_{ij,2}, T_{ij}, \phi_{ij,G}, \phi_{ij,O}, \phi_{ij,W}\right], \tag{II.2}$$

of choke opening, pressure upstream choke, pressure downstream choke, temperature, and flow composition fractions. Observations are taken at time t_{ij}, given in days since the first observation for each well. Time is used for visualization and splitting datasets. We are interested in the steady state behaviour of the flow rates. All observation are therefore averages taken over 3-9 hour intervals of stable production (Grimstad, Gunnerud, et al., 2016). Observations are shifted and scaled to lie approximately in the unit interval before model training and evaluation.

II.3.1 Data exploration

The nature of the underlying process and operational practice can create datasets that are challenging for machine learning models to deal with. For instance, it is common to see reservoir pressure decline as a well develops. As pressure declines, operators will increase the choke opening to keep flow rates stable at a given target. Some assets attempt to reduce the decline, for instance by injecting water into the reservoir (Sheng, 2014). Another remedy is to inject gas into the well flow, which makes the flow composition lighter(Guet and Ooms, 2006).

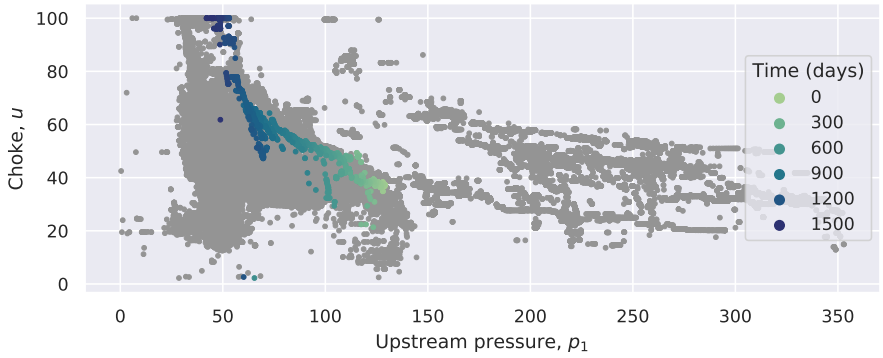


Figure II.3: Scatter plot of choke opening and upstream pressure for all wells. Observations from a single well is highlighted and coloured by days since first observation. Choke is continuously adjusted to counteract the declining reservoir pressure.

Either way, future operating points will generally not be drawn from the same distribution as the training data.

Figure II.3 illustrates the relationship between choke and pressure from all wells, with one well highlighted and colored by time. The systematic development in pressure and operational practice is clear. Models trained on such data are vulnerable to changes in operational practice. A similar pattern can be found in the flow composition, which typically develops into a higher water content with time.

All models trained on data from a single well are vulnerable to correlated explanatory variables and how data change with time. Training on data from multiple wells is one way to overcome these issues. A joint data set has several benefits. The dependencies between explanatory variables become weaker, and the variability within each explanatory variable becomes greater. This is because different wells have different operating regions and operating patterns. The reservoir development also becomes less important. Because, while a single well may move away from its previous operation region, other wells have likely operated under similar conditions before.

The joint data set contains data from 55 wells from four assets. These wells differ in design, operational practice, and reservoir conditions. Figure II.4 explore how the distribution of upstream pressure vary between wells and assets. Many wells have observations in the same range, but one asset is operating at a significantly higher pressure.

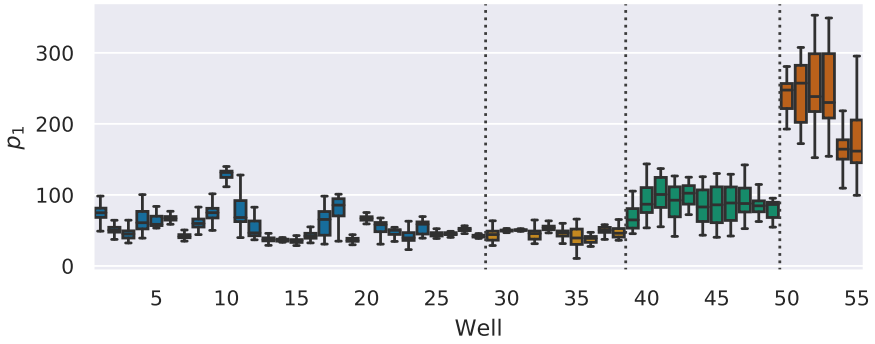


Figure II.4: Box plot of pressure upstream observations for each well. The dotted vertical lines and coloring indicate which wells are from the same asset.

II.4 Model formulation

We propose a data driven virtual flow meter with signature

$$Q_{ij} = f(x_{ij}; \gamma_j, \beta_j, \alpha). \tag{II.3}$$

It takes input variables x_{ij}, as described in Equation II.2, and is parameterized by three sets of parameters. Two of these parameter sets, γ_j and β_j, are *well specific*, while α is *shared between all wells*. The model is based on a shared neural network. The well specific parameters are used in feature adaptation and task differentiation. The model in Equation II.3 is composed of two steps. A feature adjustment step

$$z_{ij} = g(x_{ij}; \gamma_j), \tag{II.4}$$

and a flow computation step

$$Q_{ij} = h(z_{ij}; \beta_j, \alpha). \tag{II.5}$$

The composition is illustrated in Figure II.5.

II.4.1 Feature adjustment

As discussed in Section II.2.1, the observed choke opening is not directly comparable between wells. We are interested in a mapping from u for a universally comparable quantity ψ, which is analog to a CV curve. A piecewise linear mapping is chosen for this purpose. It has with m_g break points, $u_1^*, \ldots, u_{m_g}^*$, and is parametrized by $\gamma_j = \left[\gamma_{j,0}, \ldots, \gamma_{j,m_g}\right]$. It is formulated as

$$\psi_{ij} = (1 + \gamma_{j,0}) \left(u_{ij} + \sum_{k=1}^{m_g} \gamma_{j,k} \max(0, u_{ij} - u_k^*) \right), \tag{II.6}$$

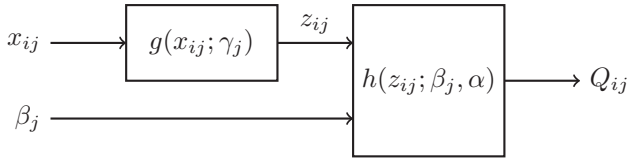


Figure II.5: Block diagram of the model architecture. The model is composed of two functions. A task specific domain adaptation g, and a flow computation h, which takes both task parameters and shared parameters.

which becomes an identity mapping if all parameters are zero. A monotonic mapping can be enforced by restricting γ_j, but this is not done here. In the examples below we use $m_g = 4$ and set breakpoints to $u_k^* = 0.2k$. Recall that u_{ij} is mapped to the unit interval.

The adjusted feature vector $z_{ij}^\top = \left[\psi_{ij}, p_{ij,1}, p_{ij,2}, T_{ij}, \phi_{ij,G}, \phi_{ij,O}\right]$ is then used to evaluate the flow computation. Note that only two of the flow composition fractions are included. This is because the fractions sum to one, and the last component is therefore redundant.

II.4.2 Flow computation

The flow rate approximation h in Equation II.5 is modelled by a residual feed forward network. The skip connection of the residual blocks spans two hidden layers with pre-activation (He et al., 2016). There are m_l layers, and all hidden layers have dimension m_h. Linear transforms are parameterized by weights $\alpha = \{(W_k, b_k)|k = 1, \ldots, m_l\}$. The rectifier function $\Phi(z_{ij}^k) = \max(0, z_{ij}^{(k)})$, where the max operation is performed elementwise, is used for activation (Goodfellow, Bengio, and Courville, 2016). There is no activation on the final layer. Adjusted features z_{ij} and task parameters β_j are stacked in a vector before the network is evaluated:

$$z_{ij}^1 = \begin{bmatrix} z_{ij} \\ \beta_j \end{bmatrix}, \tag{II.7}$$

$$z_{ij}^2 = W_1 z_{ij}^1 + b_1, \tag{II.8}$$

$$\begin{aligned} z_{ij}^{k+2} &= z_{ij}^k + W_{k+1}\Phi\left(\left[W_k\Phi\left(z_{ij}^k\right) + b_k\right]\right) + b_{k+1}, \\ k &= 2, 4, \ldots, m_l - 2, \end{aligned} \tag{II.9}$$

$$Q_{ij} = W_{m_l} z_{ij}^{m_l} + b_{m_l}. \tag{II.10}$$

The residual blocks in the neural network is illustrated in Figure II.6.

II.4.3 Model comparison

The effect of multi-task learning is explored by comparing four different models on the given data. Two conventional single-task learning models are used as

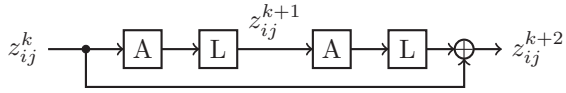


Figure II.6: Diagram of a neural network residual block as described in Equation II.9. The skip connection span two sets of activation (A) and linear layers (L).

a baseline. These are compared to two versions of the proposed multi-task architecture.

Both multi-task model formulations are identical, as described above, but they differ in how many tasks are included. The first option is trained on wells from the same asset. There are four such models because the dataset contains wells from four assets. These are referred to as "MTL-Asset" models. The second option is trained on all wells and is referred to as the "MTL-Universal" model. The two multi-task alternatives are selected to explore the degree of positive and negative transfer between tasks. These models are collectively referred to as the MTL models.

Gradient boosted trees and conventional neural networks are selected as the single-task baselines, as these represent the current state of the art. They are referred to as "STL-GBT" and "STL-ANN" respectively. Gradient boosted trees are based on the description given in (Bikmukhametov and Jäschke, 2019). The neural network models are based on the residual architecture described in Section II.4.2, but without the task parameters. Both baseline models take all observations except water fraction as input, and total flow as output. An individual copy of each baseline model is identified for each well. These models are only trained and evaluated on data from a single well.

II.5 Method

Parameters and hyperparameters are found through experimentation and optimization. The dataset is divided into development and test sets. Development data is used to identify hyperparameters and train a final set of models. Test data is only used to evaluate the performance of the final models.

II.5.1 Data splits

We split the data into subsets used for model development and testing. The development dataset is split further into training and validation sets. Data splits are visualized in Figure II.7.

Test data is selected to reflect how models are used in practice. Meaning that they are trained on all values observed up to a certain point, and then used for weeks or months before they are updated again. For each well, test data is selected such that it comes after development data in time, and the maximum distance between test and development is 120 days. The number of

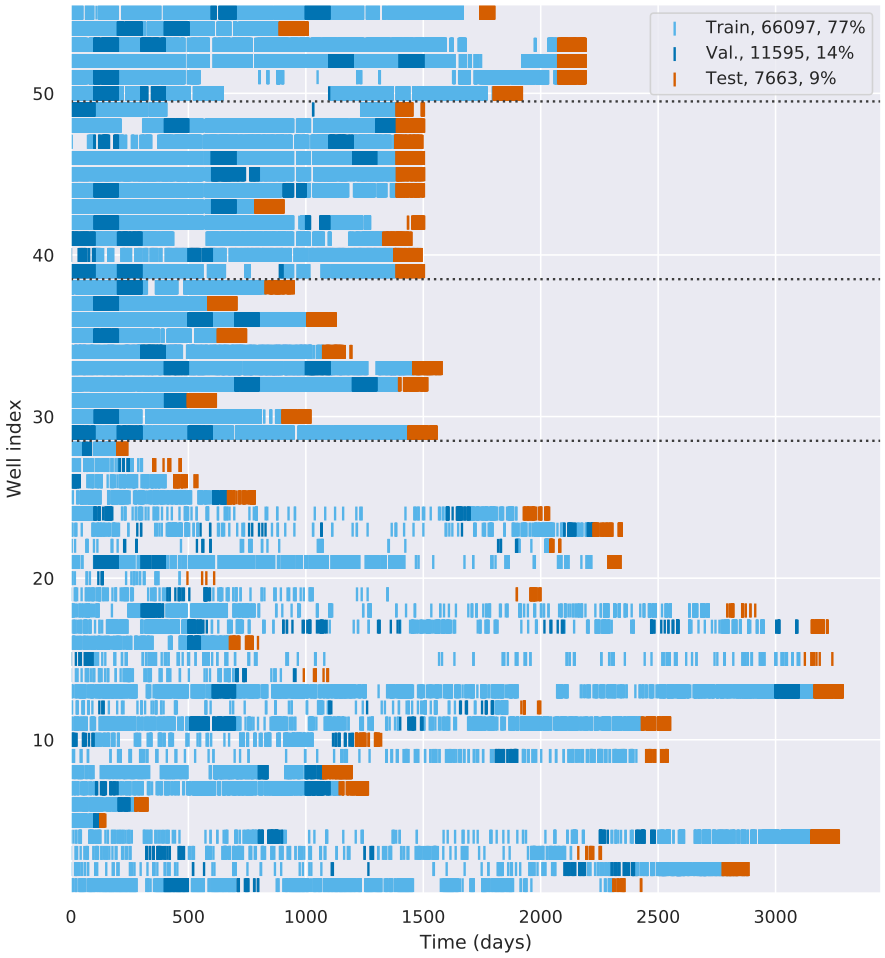


Figure II.7: Training, validation, and test data split for each well. Each mark is one data point. For some wells, the time between observations can be significant. This can be due to long periods with missing measurements, or because the well was closed. Wells from the same asset are grouped by the dotted lines.

points selected is less than 20% of the observation for that well, and less than 500.

Development data is split further into training and validation. Data points are partitioned into blocks of up to 100 consecutive days. Blocks are randomly divided between training and validation, such that the validation set is 10-20% of the total.

II.5.2 Loss and minimization

Model parameters are found by minimizing a standard loss function of prediction error and parameter regularization (Hastie, Tibshirani, and Friedman, 2009). All four model types use weighted mean square error with weights w_{ij} as prediction loss.

For the three neural network models (STL-ANN, MTL-Asset, MTL-Universal) the network parameters are regularized by a L_2 norm scaled by a factor λ. We regularize all parameters except the first neural network bias term. For the two MTL models, the task specific parameters are regularized by a L_2 norm scaled by a factor λ_T.

The loss is minimized with the AdamW optimizer (Kingma and Ba, 2015). The learning rate is set to 10^{-3}, with a decay rate of 0.5 every 100 of the last 500 epochs. Each optimization runs for 3000 epochs during hyperparameter searches. An additional 1000 epochs are used in the final training. There are three batches pr epoch. Implementation and training are done with PyTorch (Paszke et al., 2019).

STL-GBT models are regularized by penalizing the number of leaves and the squared leaf weight values (Bikmukhametov and Jäschke, 2019). Implementation and training are done with XGBoost (Chen and Guestrin, 2016).

II.5.3 Model evaluation metrics

We use absolute percentage error as the primary performance metric. For data point ij with observed flow rate Q_{ij} and predicted flow rate $\hat{Q}_{ij,M}$ from model M, we find percentage error as $e_{ij,M} = 100(\hat{Q}_{ij,M} - Q_{ij})/Q_{ij}$. For all observations we have $Q_{ij} > 0$. Model subscripts M indicate which of the four model types the error relates to, e.g., MTL-Asset. Root mean squared error is used as a secondary metric.

The mean absolute percentage error (MAPE) for well j with model M is denoted by $E_{j,M}$. Because of the heavy tails of the error distributions, we use a trimmed mean where 5% of the largest errors are removed when computing average errors for individual wells (Wilcox, 2010).

In addition to the test set performance, we will explore how the models adhere to the expected physical behavior. We expect an isolated increase in upstream pressure to increase flow rate, as indicated by Equation II.1. For any datapoint x_{ij} we have model predictions $\hat{Q}_{ij,M}$. This is compared to $\hat{Q}^{+}_{ij,M}$, which is the same model evaluated on the same data point, with the exception that $p_{ij,1}$ is increased by 10 bar. We compute a binary score

$$s_{ij,M} = \begin{cases} 0 & \text{if } \hat{Q}^{+}_{ij,M} - \hat{Q}_{ij,M} > 0, \\ 1 & \text{otherwise,} \end{cases} \tag{II.11}$$

to indicate whether this significant increase in pressure also produces an increase in flow rate. The average well score is found as $S_{j,M} = \frac{1}{N_j} \sum_{i=1}^{N_j} s_{ij,M}$. A perfect score, $S_{j,M} = 0$, corresponds to a correct sensitivity to changes in upstream pressure for all data points.

Table II.1: Summary statistics of absolute percentage error, $|e_{ij,M}|$. Statistics are computed on all test data from all wells. Reported is the mean and a set of percentiles.

Model	Mean	P05	P25	P50	P75	P95
STL-GBT	17.8	0.5	2.9	7.5	16.3	62.3
STL-ANN	20.6	0.4	2.0	4.5	11.2	44.8
MTL-Asset	10.5	0.4	1.8	4.2	9.6	42.1
MTL-Universal	12.8	0.5	2.0	4.4	8.6	33.1

II.5.4 Hyperparameter selection

Hyperparameters related to model complexity and regularization are optimized for each model individually. E.g., for the STL-ANN models we conduct 55 individual searches. Optimization is done by grid search(Bergstra and Bengio, 2012). In case multiple configurations have similar performance, the one with fewer task or network parameters is preferred.

Neural network models are controlled by the number of hidden layers m_l, hidden layer dimension m_h, and regularization factor λ. Additionally, MTL models require task parameter dimension m_β and task parameter regularization factor λ_T. These parameters are found by grid search, where candidate values are restricted based on the number of data points available for each model type.

STL-GBT models are tuned by the number of leaves, leaf weight, and the number of boosting iterations. These are all found by grid search.

Sample weight w_{ij} is set to 0.1 for multiphase meter observations and 1 for separator observations. This is motivated by the high uncertainty in multiphase meters, as discussed in Section II.1. Since most of the data is from multiphase meters, the results are not particularly sensitive to these values. These weights are used for all four model types.

II.6 Results and discussion

II.6.1 Test error overview

We first explore how the models generalize by looking at prediction errors across all wells. The results are summarized in Table II.1. The performance is quite similar for the three neural network models, but multi-task models are more robust towards large errors. The neural network models outperform STL-GBT. The distribution of prediction errors is heavy tailed, with a few outliers skewing the mean errors. These outliers motivate the use of trimmed mean when results are reported on a well by well basis.

Figure II.8 illustrates how prediction errors develop with time. As expected, the performance degrades with time for all model types. All model types have great performance in the first few weeks. The benefit of multi-task learning becomes apparent after six weeks.

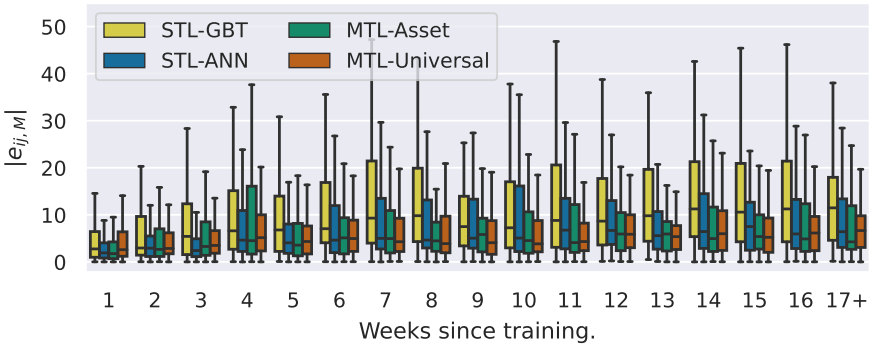


Figure II.8: Box plot of absolute percentage error, grouped by weeks since the last training datapoint. Errors are computed on all test data for all wells.

Table II.2: Mean prediction errors are computed as trimmed MAPE and RMSE for each well. This yields 55 error estimates for each combination of model and metric, which are summarized by their mean and a set of percentiles.

Metric	Model	Mean	P05	P25	P50	P75	P95
$E_{j,M}^{MAPE}$	STL-GBT	14.5	2.3	5.8	8.6	10.8	53.9
	STL-ANN	10.4	1.4	3.8	5.7	11.1	34.5
	MTL-Asset	8.2	1.4	3.5	6.2	9.0	22.2
	MTL-Universal	7.5	1.6	3.5	5.0	9.2	19.8
$E_{j,M}^{RMSE}$	STL-GBT	9.6	2.0	3.7	6.3	11.0	27.0
	STL-ANN	7.2	1.2	2.5	4.1	8.9	22.4
	MTL-Asset	5.7	1.2	2.5	4.1	7.5	14.1
	MTL-Universal	5.5	0.8	2.5	3.7	6.9	17.1

II.6.2 Well by well performance

Wells have a different number of data points, and the errors reported in Section II.6.1 will naturally be dominated by the wells with many data points. We now explore the test set errors for individual wells. Trimmed MAPE and RMSE values for each well is summarized in Table II.2. The three neural network models have similar performance for the best half of the wells, with performance comparable to conventional multiphase meters. Multi-task models are significantly better on the more challenging wells. Neural network models generally outperform STL-GBT. MAPE and RMSE reveal similar patterns. The remainder of the analysis will focus on MAPE values.

Table II.3: Model performance grouped by assets. Reported is the average trimmed MAPE for wells from the same asset, for the four model types. The best model type is highlighted.

Model	A. 1	A. 2	A. 3	A. 4
STL-GBT	15.6	13.5	10.4	18.3
STL-ANN	10.9	13.8	5.9	10.5
MTL-Asset	8.1	**10.2**	6.5	7.9
MTL-Universal	**7.3**	11.3	**5.7**	**4.9**

II.6.3 Asset performance

The two MTL models are trained on wells from the same asset and on all wells. To explore the degree of positive and negative knowledge transfer, performance is explored on the four assets. The results are summarized in in Table II.3. Apart from Asset 3, which has great performance for all neural network models, there is a clear benefit of shared data. For assets 1, 2, and 4, the best multi-task model offers a 25-50% error reduction compared to STL-ANN. However, it is an open question to decide which level of data sharing is best suited for a given well or asset. All model types struggle with Asset 2, which could be due to the limited excitation seen in Figure II.4. Apart from Asset 2, MTL model performance is close to that expected from conventional multiphase meters.

II.6.4 Sensitivity analysis

Models with great test set performance can still suffer from the data challenges presented in Section II.3.1. Correlated explanatory variables make it difficult to isolate the effect of individual variables. Figure II.9 illustrates the issue for Well 7. We expect the response to an increase in upstream pressure to be an increase in flow rate, as indicated by the mechanistic model in Equation II.1. For well 7, both STL-ANN and MTL-Universal models have low test errors, with trimmed MAPE being 2.2% and 1.6% respectively. There is however a significant difference in how they have interpreted the explanatory variables. In this case, the MTL-Universal model was able to identify the expected response, while the STL-ANN was not.

The observations from Figure II.9 are generalized using the sensitivity metric described in Section II.5.3. The results are given in Table II.4. A large majority of wells remain unchanged or achieve a better sensitivity score by sharing data with other wells. The advantage of transfer learning is clear in this comparison.

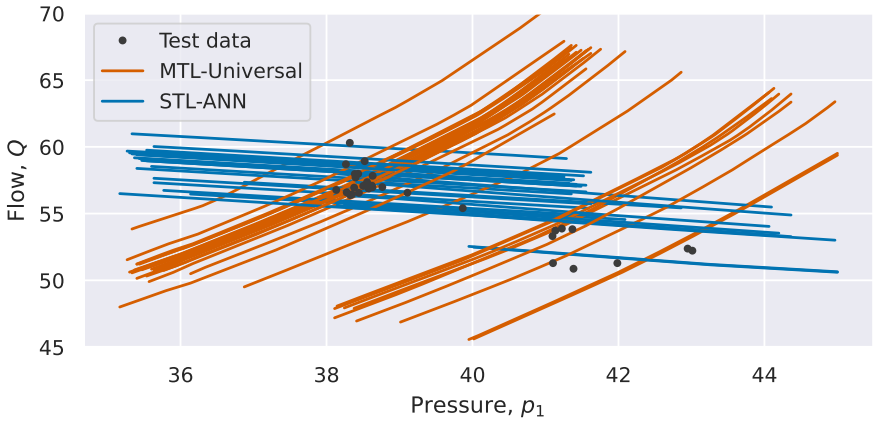


Figure II.9: Comparison of model sensitivity for well 7. Each model is evaluated by taking a subset of 30 test data points (black dots) and varying upstream pressure in a neighborhood around the observed value. The response of STL-ANN is given in blue and MTL-Universal in orange.

Table II.4: Mean sensitivity error $S_{j,M}$ for the four model types. Zero is the best score, which means a model had the correct sensitivity for each data point for a given well.

Model	Error
STL-GBT	0.56
STL-ANN	0.28
MTL-Asset	0.14
MTL-Universal	0.07

II.6.5 Ablation study

An ablation study is conducted to better understand the improvements seen in the proposed architecture (Lipton and Steinhardt, 2019). The proposed model extends the current state-of-the-art with multi-task learning. Two task adaptation mechanisms are included. These are task parameters β_j, and domain adaptation parameterized by γ_j. To explore the effect of the task adaptation, the MTL-Universal model is trained with one or both elements removed. When both elements are removed, the model is reduced to a single-task neural network trained on data from all wells. New hyperparameters are found for each ablation. The results is given in Table II.5. On average, both adaptation mechanisms have a similar impact. The inclusion of both is beneficial for overall performance, but with diminishing returns, as they are potentially overlapping.

Table II.5: Summary of ablations conducted on the MTL-Universal model. Average trimmed MAPE is computed on all wells. Value for the complete model is repeated from Table II.2.

Ablation	Error
Remove γ_j and β_j	10.5
Remove β_j	8.8
Remove γ_j	8.4
Complete model	7.5

II.6.6 Model complexity

A multi-task model is more complex than an isolated single-task model. However, as the number of tasks grows, there are several aspects to multi-task learning that leads to overall less complexity. Table II.6 summarizes the number of parameters and training time for the four model types presented. STL-GBT is a separate class of models and is much faster to compute than neural networks. For neural network models, the larger models require more time for each model, but less time overall.

In the universal model, the number of well parameters is significantly smaller than the number of parameters needed in an individual well model. On average, an MTL-Asset model requires almost the same number of parameters as MTL-Universal. Indicating that model size does not need to grow significantly with the number of tasks.

Table II.6: Summary of model complexity, judged by the number of parameters and time required to train models for all wells. E.g., it took 21 minutes and 25 seconds to train the four MTL-Asset models, and they have 711389 parameters in total. All models are trained on a single GPU.

Model	Models	Time	Parameters
STL-GBT	55	00:57	-
STL-ANN	55	56:06	450455
MTL-Asset	4	21:25	711389
MTL-Universal	1	12:38	203851

In terms of manual work and maintenance, the MTL-Universal formulation scales better with additional wells than conventional model formulations, because it is only one neural network that must be curated. This is highly advantageous for the practical application and commercialization of the results.

II.6.7 Qualitative properties

The selected hyperparameter configuration has two task parameters for each well, $\beta_j^T = [\beta_{j,1}, \beta_{j,2},]$. Figure II.10 illustrates the identified parameters for all wells. Asset 3 has two types off wells, oil producers and gas producers, which

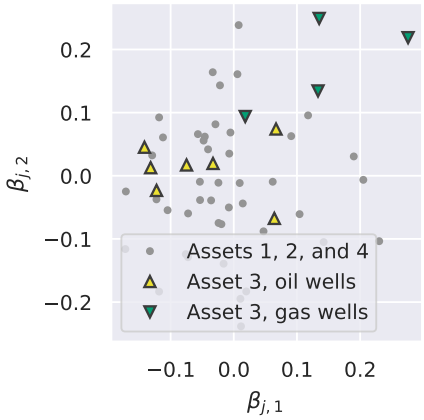


Figure II.10: Plot of β_j from the universal model. Asset 3 has two distinct classes of wells, oil producers and gas producers, which are highlighted by triangle markers.

have different choke geometries and fluid properties. These wells are separated in space according to their classes, which indicates that task parameters capture physical properties, rather than being proxies for the well index. To further support this, Figure II.11 illustrates how changes in task parameters alter the shape and magnitude of the model response in a consistent fashion.

II.7 Conclusion

A multi-task learning architecture for data driven virtual flow metering was proposed and explored in a study of 55 wells from four assets. The proposed architecture successfully addresses the identified data challenges, while generally improving model performance. Sharing data between wells improves robustness towards changes in operational practice and makes the model adhere better to the expected physical relationships. In terms of prediction errors, all assets benefit from some level of data sharing. Two of the assets saw average errors reduced by 30–50% when data was shared between all assets. One asset saw a reduction of 24% when data was shared within the asset, but only 16% improvement when data is shared between all assets. This indicates that issues related to negative transfer could be present in the VFM problem. The final asset saw no significant improvements because the single task architectures already performed well. Overall, the MTL architecture is a promising step towards a data driven virtual flow meter solution.

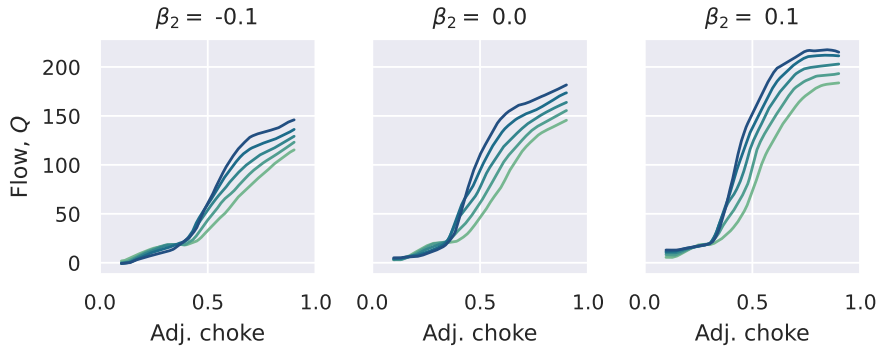


Figure II.11: Effect of β_j on model predictions. The universal model is evaluated on a fixed operating point, except for varying the adjusted choke between 0.1 and 0.9. Each subplot has a fixed value for $\beta_{j,2}$, given in the title. Each curve has a fixed value for $\beta_{j,1}$, ranging from -0.1 for the light green to 0.1 for the dark blue.

II.7.1 Future work

All wells explored here have many data points. It is expected that wells with fewer observations will see a greater benefit from the knowledge sharing architecture. Exploring these opportunities is left as future work. Additionally, it is desirable to further explore task synergies and negative transfer in the VFM context.

In the proposed model, task specific parameters are constants. In practice, these are likely time varying, since both the well and reservoir will develop over time, e.g., changes in fluid properties, or equipment wear and tear. These aspects are topics for future research.

Acknowledgements

This work was supported by Solution Seeker Inc. and The Research Council of Norway.

References

AIME and Society of Petroleum Engineers (1984). *The SI Metric System of Units and SPE Metric Standard*. Society of Petroleum Engineers.

Amin, A. (2015). "Evaluation of Commercially Available Virtual Flow Meters (VFMs)". In: *Proceedings of the Annual Offshore Technology Conference*, pp. 1293–1318.

Bergstra, J. and Bengio, Y. (2012). "Random search for hyper-parameter optimization". In: *Journal of Machine Learning Research* vol. 13, pp. 281–305.

Bikmukhametov, T. and Jäschke, J. (2019). "Oil Production Monitoring Using Gradient Boosting Machine Learning Algorithm". In: *12th IFAC Symposium on Dynamics and Control of Process Systems, including Biosystems*. Vol. 52. IFAC-PapersOnLine, pp. 514–519.

— (2020). "First Principles and Machine Learning Virtual Flow Metering: A Literature Review". In: *Journal of Petroleum Science and Engineering* vol. 184, no. September 2019, p. 106487.

Caruana, R. (1997). "Multitask Learning". In: *Machine Learning* vol. 28, no. 1, pp. 41–75.

Chen, T. and Guestrin, C. (2016). "XGBoost: A Scalable Tree Boosting System". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '16. San Francisco, California, USA, pp. 785–794.

Corneliussen, S. et al. (2005). *Handbook of multiphase flow metering*. 2nd ed. Norwegian Society for Oil and Gas Measurement.

Dorado-Moreno, M. et al. (2020). "Multi-task learning for the prediction of wind power ramp events with deep neural networks". In: *Neural Networks* vol. 123, pp. 401–411.

Foss, B., Knudsen, B. R., and Grimstad, B. (2018). "Petroleum production optimization – A static or dynamic problem?" In: *Computers & Chemical Engineering* vol. 114, pp. 245–253.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press.

Grace, A. and Frawley, P. (2011). "Experimental parametric equation for the prediction of valve coefficient (Cv) for choke valve trims". In: *International Journal of Pressure Vessels and Piping* vol. 88, no. 2-3, pp. 109–118.

Grimstad, B., Gunnerud, V., et al. (2016). "A Simple Data-Driven Approach to Production Estimation and Optimization". In: *SPE Intelligent Energy International Conference and Exhibition*.

Grimstad, B., Hotvedt, M., et al. (2021). "Bayesian neural networks for virtual flow metering: An empirical study". In: *Applied Soft Computing* vol. 112, p. 107776.

Guet, S. and Ooms, G. (2006). "Fluid mechanical aspects of the gas-lift technique". In: *Annual Review of Fluid Mechanics* vol. 38, no. 1, pp. 225–249.

Hannun, A. Y. et al. (2019). "Cardiologist-level arrhythmia detection and classification in ambulatory electrocardiograms using a deep neural network". In: *Nature Medicine* vol. 25, no. 1, pp. 65–69.

Hansen, L. S., Pedersen, S., and Durdevic, P. (2019). "Multi-phase flow metering in offshore oil and gas transportation pipelines: Trends and perspectives". In: *Sensors* vol. 19, no. 9.

Hastie, T., Tibshirani, R., and Friedman, J. H. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. 2nd ed. Springer series in statistics. Springer.

He, K. et al. (2016). "Identity Mappings in Deep Residual Networks". In: *Computer Vision – ECCV 2016*. Cham, pp. 630–645.

Hong, C., Yu, J., Wan, J., et al. (2015). “Multimodal Deep Autoencoder for Human Pose Recovery”. In: *IEEE Transactions on Image Processing* vol. 24, no. 12, pp. 5659–5670.

Hong, C., Yu, J., Zhang, J., et al. (2019). “Multimodal Face-Pose Estimation With Multitask Manifold Deep Learning”. In: *IEEE Transactions on Industrial Informatics* vol. 15, no. 7, pp. 3952–3961.

Jamaluddin, A. K. and Kabir, C. S. (2012). “Flow assurance: Managing flow dynamics and production chemistry”. In: *Journal of Petroleum Science and Engineering* vol. 100, pp. 106–116.

Jin, N. et al. (2020). “Multi-task learning model based on multi-scale CNN and LSTM for sentiment classification”. In: *IEEE Access* vol. 8, pp. 77060–77072.

Kanshio, S. (2020). “A review of hydrocarbon allocation methods in the upstream oil and gas industry”. In: *Journal of Petroleum Science and Engineering* vol. 184.

Kingma, D. P. and Ba, J. L. (2015). “Adam: A method for stochastic optimization”. In: *3rd International Conference on Learning Representations*. San Diego.

Lipton, Z. C. and Steinhardt, J. (2019). “Troubling Trends in Machine Learning Scholarship: Some ML Papers Suffer from Flaws That Could Mislead the Public and Stymie Future Research.” In: *Queue* vol. 17, no. 1, pp. 45–77.

Lu, J. et al. (2015). “Transfer learning using computational intelligence: A survey”. In: *Knowledge-Based Systems* vol. 80, pp. 14–23.

Majumder, N. et al. (2019). “Sentiment and Sarcasm Classification With Multitask Learning”. In: *IEEE Intelligent Systems* vol. 34, no. 3, pp. 38–43.

Mei, B. and Xu, Y. (2020a). “Multi-task ν-twin support vector machines”. In: *Neural Computing and Applications* vol. 32, no. 15, pp. 11329–11342.

— (2020b). “Safe sample screening for regularized multi-task learning”. In: *Knowledge-Based Systems* vol. 204, p. 106248.

Misra, I. et al. (2016). “Cross-Stitch Networks for Multi-task Learning”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3994–4003.

Paszke, A. et al. (2019). “PyTorch: An imperative style, high-performance deep learning library”. In: *Advances in Neural Information Processing Systems* vol. 32.

AL-Qutami, T. A., Ibrahim, R., and Ismail, I. (2017). “Hybrid neural network and regression tree ensemble pruned by simulated annealing for virtual flow metering application”. In: *2017 IEEE International Conference on Signal and Image Processing Applications (ICSIPA)*, pp. 304–309.

AL-Qutami, T. A., Ibrahim, R., Ismail, I., and Ishak, M. A. (2017). “Development of soft sensor to estimate multiphase flow rates using neural networks and early stopping”. In: *International Journal on Smart Sensing and Intelligent Systems* vol. 10, no. 1, pp. 199–222.

— (2018). “Virtual multiphase flow metering using diverse neural network ensemble and adaptive simulated annealing”. In: *Expert Systems with Applications* vol. 93, pp. 72–85.

Schüller, R. B., Solbakken, T., and Selmer-Olsen, S. (Aug. 2003). "Evaluation of multiphase flow rate models for chokes under subcritical oil/gas/water flow conditions". In: *SPE Production and Facilities* vol. 18, no. 3, pp. 170–181.

Sheng, J. J. (2014). "Critical review of low-salinity waterflooding". In: *Journal of Petroleum Science and Engineering* vol. 120, pp. 216–224.

Sigtia, S. et al. (2020). "Multi-Task Learning for Speaker Verification and Voice Trigger Detection". In: *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Barcelona, Spain.

Silver, D. L., Poirier, R., and Currie, D. (2008). "Inductive transfer with context-sensitive neural networks". In: *Machine Learning* vol. 73, no. 3, pp. 313–336.

Standley, T. et al. (2020). "Which Tasks Should Be Learned Together in Multi-task Learning?" In: *Proceedings of the 37th International Conference on Machine Learning*. Vol. 119. Proceedings of Machine Learning Research, pp. 9120–9132.

Stenhouse, B. J. (2008). "Modelling and optimisation in BP E&P". In: *SPE Intelligent Energy Conference & Exhibition*. Vol. 2. Amsterdam, The Netherlands, pp. 638–645.

Thorn, R., Johansen, G. A., and Hjertaker, B. T. (2012). "Three-phase flow measurement in the petroleum industry". In: *Measurement Science and Technology* vol. 24, no. 1.

Vaswani, A. et al. (2017). "Attention is All you Need". In: *Advances in Neural Information Processing Systems*. Vol. 30. Long Beach, CA.

White, F. M. (2008). *Fluid Mechanics*. 6, revised. McGraw-Hill series in mechanical engineering. McGraw Hill.

Wilcox, R. R. (2010). *Fundamentals of Modern Statistical Methods: Substantially Improving Power and Accuracy*. 2nd ed. Springer.

Wu, Z., Li, Q., and Xia, X. (2021). "Multi-timescale Forecast of Solar Irradiance Based on Multi-task Learning and Echo State Network Approaches". In: *IEEE Transactions on Industrial Informatics* vol. 17, no. 1, pp. 300–310.

Yu, J. et al. (2019). "Hierarchical Deep Click Feature Prediction for Fine-grained Image Recognition". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Zhang, Y. and Yang, Q. (2021). "A Survey on Multi-Task Learning". In: *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–20.

Zhou, Y. et al. (2021). "A novel combined multi-task learning and Gaussian process regression model for the prediction of multi-timescale and multi-component of solar radiation". In: *Journal of Cleaner Production* vol. 284, p. 124710.

Zintgraf, L. et al. (2019). "Fast Context Adaptation via Meta-Learning". In: *Proceedings of the 36th International Conference on Machine Learning*. Vol. 97. Proceedings of Machine Learning Research, pp. 7693–7702.