Incomplete Conditional Density Estimation for Fast Materials Discovery

Phuoc Nguyen, Truyen Tran, Sunil Gupta, Santu Rana and Svetha Venkatesh
Applied AI Institute, Deakin University, Geelong, Australia
{phuoc.nguyen,truyen.tran,sunil.gupta,santu.rana,svetha.venkatesh}@deakin.edu.au

Abstract

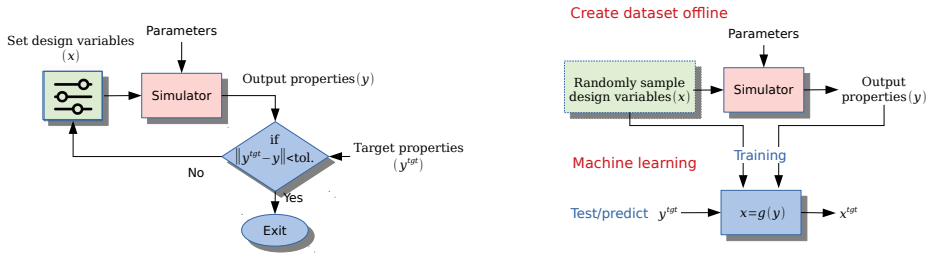
Discovering new physical products and processes often demands enormous experimentation and expensive simulation. To design a new product with certain target characteristics, an extensive search is performed in the design space by trying out a large number of design combinations before reaching to the target characteristics. However, forward searching for the target design becomes prohibitive when the target is itself moving or only partially understood. To address this bottleneck, we propose to use backward prediction by leveraging the rich data generated during earlier exploration and construct a machine learning framework to predict the design parameters for any target in a single step. This poses two technical challenges: the first caused due to one-to-many mapping when learning the inverse problem and the second caused due to an user specifying the target specifications only partially. To overcome the challenges, we formulate this problem as conditional density estimation under high-dimensional setting with incomplete input and multimodal output. We solve the problem through a deep hybrid generative-discriminative model, which is trained end-to-end to predict the optimum design.

1 Introduction

Scientific innovations relating physical processes require laborious experimentation and expensive simulation as the relationships between design variables and output characteristics are unknown [2]. To design a new product with certain target characteristics, a search is typically performed in the design space – a large number of the design combinations (input variables) are tried in simulators before reaching to the target characteristics (see Fig. 1a). Although modern search methods such as Bayesian Optimization are efficient, there is an inherent problem with the search: Each time the target characteristics are changed or refined, the search process needs to be restarted making the design task extremely time-consuming. The current search paradigm does not harness existing experimentation data and simulator queries systematically as there is no clear provision to reuse them.

Thanks to the availability of growing data, accurate simulators and modern machine learning algorithms, this search process can be avoided with a potential to accelerate scientific innovations by multiple orders of magnitude. An effective way to address this problem is to leverage the existing data and query the simulators in an offline mode to sample the data space sufficiently and then harness this data to build a machine learning model. Given sufficient data, modern machine learning (ML) models (e.g. a deep neural network) can approximate the underlying physical relationships and the simulators arbitrarily closely. The ML models can then be used to convert the search process in a less expensive optimization. Denoting the function learned by ML model as $f(\boldsymbol{x})$, we can discover the target design by solving the following optimization problem: $\boldsymbol{x}^{\text{target}} = \underset{\boldsymbol{x}\in\mathcal{X}}{\operatorname{argmin}} \|f(\boldsymbol{x}) - \boldsymbol{y}^{\text{target}}\|$.

Preprint. Work in progress.



(a) Search process for simulation-based design. It may run for hours or days per one design variable set.

(b) Machine learning approach, typically costs miniseconds per tens of predictions.

Figure 1: Experimental design paradigms for design and discovery of new products. (a) current design paradigm requiring a long iterative search procedure (b) The proposed design paradigm.

Although the above–mentioned approach can replace the expensive search via experimentation and simulators, we still need to solve a global optimization problem each time we have a specified design goal $\boldsymbol{y}^{\text{target}}$. Fortunately, we can also avoid the global optimization by simply flipping our problem of learning $f(\boldsymbol{x})$ to the learning of its *inverse design function* $g(\boldsymbol{x}) = f^{-1}(\boldsymbol{x})$. This offers a paradigm shift in modeling as by learning the function $\boldsymbol{x} = g(\boldsymbol{y})$, we can directly predict our design variables in a single step as $\boldsymbol{x}^{\text{target}} = g(\boldsymbol{y}^{\text{target}})$ without needing any search or global optimization (see Fig. 1b).

However when taking this approach to designing new products, we face two technical difficulties. The first difficulty is that the forward function f may be many-to-one in some problem domains, *i.e.* it might be possible to have same output for many input combinations. In such cases, the inverse function will be ill-posed as there would be multiple $\boldsymbol{x}$ solutions for the same $\boldsymbol{y}$. Another practical difficulty that arises when designing new product and processes is partial specification of the target characteristics. This happens mainly for two reasons: the first, an user may not be absolutely certain about setting all the target characteristics, instead he/she might want to leave some of the target characteristics free; and the second, he/she may not know the precise value of a certain target characteristic. Due to these $\boldsymbol{y}^{\text{target}}$ may only be partially specified for a new product.

We overcome the difficulties by formulating this inverse design problem as an instance of conditional density estimation under high-dimensional setting with incomplete input and multimodal output. We solve the problem by designing a deep hybrid generative-discriminative model, which integrates Mixture Density Networks (MDN) [1] and Conditional Variational Autoencoder (CVAE) [5, 15]. The MDN component addresses the first difficulty allowing multiple designs for a target. The CVAE component addresses the second difficulty by imputing the missing target specifications through the data distribution. Putting together in the integrated model permits multiple design possibilities catering for the degree of freedom caused by either the many-to-one forward function and the incomplete specification. As a result, we gain the freedom to discover a whole sub-class of products meeting the partial target specifications.

We focus on an alloy design problem and use a metallurgical model known as CALPHAD implemented in the Thermo-Calc software [4, 9]. The input to our model is a partial phase diagram and the output is an alloy composition. Our goal is to specify a desired phase diagram that is associated to different alloy properties, e.g., its strength, weld-ability, its corrosion resistance etc., and predict an alloy elemental composition that is highly likely to form the specified phases and therefore show the intended properties. We carried out experiments on two datasets acquired by querying the Aluminum alloy database. The first dataset was created from 30 Aluminum alloys[1]. The second dataset was created by using Bayesian Optimization and searching for alloys satisfying a common FCC property existed in the 30 mentioned Aluminum alloys. We vary the composition of each alloy in both sets in a window of $\pm 20\%$ to derive necessary data to train and test our models. We show that our model can accurately predict the alloy compositions with an average relative error of 1.8% and 2.95% for the first and second datasets respectively.

[1]Alloy IDs: 2014, 2018, 2024, 2025, 2218, 2219, 2618, 6053, 6061, 6063, 6066, 6070, 6082, 6101, 6151, 6201, 6351, 6463, 6951, 7001, 7005, 7020, 7034, 7039, 7068, 7075, 7076, 7175, 7178, 7475.

2 Methods

2.1 Inverse-problem as data–driven inference

Let $\boldsymbol{x} \in \mathcal{X}$ be a vector describing the design parameters, and $\boldsymbol{y}$ be a set of target properties. As an example, in alloy discovery, the design parameters can be the mixture of components of the alloy, and the target properties can be melting temperature, hardness, elasticity and surface resistance against corrosion. Physical laws dictate that there exist a function of the form $\boldsymbol{y} = f(\boldsymbol{x})$. As simulation of realistic matters is extremely complex[2], accurate simulation of $f(\boldsymbol{x})$ often demands great computational and time resources *for each* $\boldsymbol{x}$. This may prevent comprehensive exploration of the huge design space to reach a desirable target.

Machine learning offers an alternative data–driven approach to vastly speed up this exploration. On one hand, we can approximate the function $f(\boldsymbol{x})$ by learning a variational fast alternative $f_\pi(\boldsymbol{x})$ parametrized by π (e.g., weights of a deep neural net). Learning occurs only once on a training set $D = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}$, where $\boldsymbol{x}_i$ is a sample in $\mathcal{X}$ and $\boldsymbol{y}_i$ is computed by running the simulator. Once the function has been estimated, the search of a desirable target $\boldsymbol{y}^{target}$ can be efficiently carried out through global optimization $\boldsymbol{x}^{target} = \text{argmin}_{\boldsymbol{x}\in\mathcal{X}} \|f_\pi(\boldsymbol{x}) - \boldsymbol{y}^{\text{target}}\|$. On the other hand, we can eliminate the global optimization entirely by estimating an *inverse-function* of f using $g_\eta(\boldsymbol{y})$ such as that $g_\eta(\boldsymbol{y}_i) \approx \boldsymbol{x}_i$ for all i. With this inverse-function, searching for target designs $\boldsymbol{x}^{target}$ to meet target properties $\boldsymbol{y}^{target}$ is instant.

However, estimating $g_\eta(\boldsymbol{y})$ is challenging for the following reasons. First, the inverse function f^{-1} may not exist since f can be a many-to-one mapping, that is, one target output $\boldsymbol{y}$ can be satisfied by multiple input designs $\boldsymbol{x}$. Second, in practice the target output may not be fully specified by the scientist. Sometimes this incomplete specification can be due to ignorant (we do not know exactly what we want in the first place), or on purpose (we want not an exact design, but a class of designs). A partial specification of $\boldsymbol{y}$ further increases the uncertainty about the input design $\boldsymbol{x}$. Let $\boldsymbol{y} = (\boldsymbol{v}, \boldsymbol{h})$ where $\boldsymbol{v}$ denotes the specified target component and $\boldsymbol{h}$ the unspecified counterpart. Given the uncertainty in the target designs $\boldsymbol{x}$, *the inverse-problem is best reformulated as estimating an entire conditional density spectrum* $P(\boldsymbol{x} \mid \boldsymbol{v})$.

2.2 Multimodal density estimation given incomplete conditions

Assume that $\boldsymbol{h}$ is generated by a latent variable $\boldsymbol{z}$. We aim to model a conditional density of the design $\boldsymbol{x}$ given the specified $\boldsymbol{v}$ target component:

$$P(\boldsymbol{x} \mid \boldsymbol{v}) = \int_{\boldsymbol{h}} \int_{\boldsymbol{z}} P(\boldsymbol{x}, \boldsymbol{h}, \boldsymbol{z} \mid \boldsymbol{v})\, d\boldsymbol{h} d\boldsymbol{z} \tag{1}$$

The joint density $P(\boldsymbol{x}, \boldsymbol{h}, \boldsymbol{z} \mid \boldsymbol{v})$ is further factorized as:

$$P(\boldsymbol{x}, \boldsymbol{h}, \boldsymbol{z} \mid \boldsymbol{v}) = P(\boldsymbol{x} \mid \boldsymbol{v}, \boldsymbol{h})\, P(\boldsymbol{h} \mid \boldsymbol{v}, \boldsymbol{z})\, P(\boldsymbol{z} \mid \boldsymbol{v}) \tag{2}$$

This leads to

$$\begin{aligned} P(\boldsymbol{x} \mid \boldsymbol{v}) &= \int_{\boldsymbol{h}} \int_{\boldsymbol{z}} P(\boldsymbol{x} \mid \boldsymbol{v}, \boldsymbol{h})\, P(\boldsymbol{h} \mid \boldsymbol{v}, \boldsymbol{z})\, P(\boldsymbol{z} \mid \boldsymbol{v}) d\boldsymbol{h} d\boldsymbol{z} \\ &\approx \frac{1}{N} \sum_{i=1}^{N} \mathbb{E}_{P\left(\boldsymbol{h} \mid \boldsymbol{v}, \boldsymbol{z}^{(i)}\right)} \left[P(\boldsymbol{x} \mid \boldsymbol{v}, \boldsymbol{h})\right] \end{aligned} \tag{3}$$

where $\boldsymbol{z} \sim P(\boldsymbol{z} \mid \boldsymbol{v})$.

However, integrating over $\boldsymbol{h}$ is still intractable. Assume further that the imputation of the missing component via $P(\boldsymbol{h} \mid \boldsymbol{v}, \boldsymbol{z})$ takes a simple Gaussian form, that is, $\boldsymbol{h} \sim \mathcal{N}\left(\boldsymbol{\mu}(\boldsymbol{v}, \boldsymbol{z}); I\sigma^2(\boldsymbol{v}, \boldsymbol{z})\right)$. For simplicity, we approximate the expectation $\mathbb{E}_{P\left(\boldsymbol{h} \mid \boldsymbol{v}, \boldsymbol{z}^{(i)}\right)} \left[f(\boldsymbol{h})\right]$ by a function evaluation at the mode, i.e., $\mathbb{E}_{P\left(\boldsymbol{h} \mid \boldsymbol{v}, \boldsymbol{z}^{(i)}\right)} \left[f(\boldsymbol{h})\right] \approx f\left(\bar{\boldsymbol{h}}\right)$ where $\bar{\boldsymbol{h}} = \boldsymbol{\mu}(\boldsymbol{v}, \boldsymbol{z})$. This leads to

$$P(\boldsymbol{x} \mid \boldsymbol{v}) \approx \frac{1}{N} \sum_{i=1}^{N} P\left(\boldsymbol{x} \mid \boldsymbol{v}, \boldsymbol{\mu}\left(\boldsymbol{v}, \boldsymbol{z}^{(i)}\right)\right) \tag{4}$$

[2]Quantum computation using DFT takes hours to compute chemical properties of small molecules, but it is practically impossible to compute for just one micro-cube of matters even using the best supercomputer.

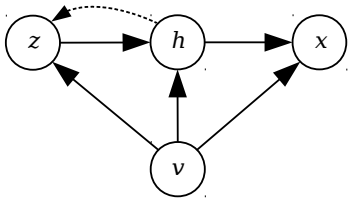


Figure 2: Graphical model of the proposed method. $\boldsymbol{x}$: input design, $\boldsymbol{v}$: specified target component, $\boldsymbol{h}$: unspecified part, and $\boldsymbol{z}$: latent variable.

As $\boldsymbol{\mu}\left(\boldsymbol{v}, \boldsymbol{z}^{(i)}\right)$ serves as a reconstruction of the missing component $\boldsymbol{h}$, this somewhat resembles the technique of multiple imputations in the literature [13]. The graphical model of the factorization in Eq. (2) is depicted in Fig. 2.

2.2.1 Multimodal density

Due to the one-to-many mapping in the inverse problem, the conditional density $P\left(\boldsymbol{x} \mid \boldsymbol{v}, \bar{\boldsymbol{h}}\right)$ may be multimodal, that is, we may have multiple classes of solution. For example, in the case of alloy design, this is naturally the case as alloys seem to concentrate around rather separate modes each of which has a dominant metal. To account for multiple modes, we propose to use the mixture model:

$$P_\gamma\left(\boldsymbol{x} \mid \boldsymbol{v}, \bar{\boldsymbol{h}}\right) = \sum_{k=1}^{K} \alpha_k P_k\left(\boldsymbol{x} \mid \boldsymbol{v}, \bar{\boldsymbol{h}}\right) \tag{5}$$

with mixture components α_k subject to $\alpha_k \geq 0$ and $\sum_{k=1}^{K} \alpha_k = 1$. The mixture components are implemented using a softmax neural network, i.e., $\alpha_k = \text{softmax}\left(f_k\left(\boldsymbol{v}, \bar{\boldsymbol{h}}\right)\right)$ where f_k is a neural network.

Using the re-parametrization trick (e.g., see [5]) we model $P_k\left(\boldsymbol{x} \mid \boldsymbol{y}\right)$ as a Gaussian of mean $\boldsymbol{\mu}_k = g_k^\mu\left(\boldsymbol{v}, \bar{\boldsymbol{h}}\right)$ and isotropic covariance matrix $I\sigma_k^2$ for identity matrix I and $\sigma_k^2 = \exp\left(g_k^\sigma\left(\boldsymbol{v}, \bar{\boldsymbol{h}}\right)\right)$. Here $g_k^\mu\left(\boldsymbol{v}, \bar{\boldsymbol{h}}\right)$ and $g_k^\sigma\left(\boldsymbol{v}, \bar{\boldsymbol{h}}\right)$ are parametrized as deep neural networks, making this model resemble the mixture density network (MDN) [1].

2.3 Hybrid discriminative-generative learning

Now what remains is a model to account for the generation of the missing target $P\left(\boldsymbol{h} \mid \boldsymbol{v}\right) = \int_{\boldsymbol{z}} P\left(\boldsymbol{h} \mid \boldsymbol{v}, \boldsymbol{z}\right) P\left(\boldsymbol{z} \mid \boldsymbol{v}\right) d\boldsymbol{z}$. Adapting the formulation of the Conditional Variational AutoEncoder (CVAE) [5, 15], the evidence lower bound (ELBO) is maximized:

$$\begin{aligned}\mathcal{L}_{\text{CVAE}}\left(\theta, \phi\right) = &-D_{\text{KL}}\left(Q_\phi\left(\boldsymbol{z} \mid \boldsymbol{h}, \boldsymbol{v}\right) \| P\left(\boldsymbol{z}\right)\right) + \\ &+ \mathbb{E}_{z \sim Q_\phi}\left[\log P_\theta\left(\boldsymbol{h} \mid \boldsymbol{z}, \boldsymbol{v}\right)\right]\end{aligned} \tag{6}$$

where $Q_\phi\left(\boldsymbol{z} \mid \boldsymbol{h}, \boldsymbol{v}\right)$ denotes the recognition model that approximates the posterior $P\left(\boldsymbol{z} \mid \boldsymbol{h}, \boldsymbol{v}\right)$. More precisely $Q_\phi\left(\boldsymbol{z} \mid \boldsymbol{h}, \boldsymbol{v}\right)$ is a multivariate Gaussian of mean $\boldsymbol{\mu}_\phi\left(\boldsymbol{h}, \boldsymbol{v}\right)$ and diagonal covariance $I\sigma_\phi^2\left(\boldsymbol{h}, \boldsymbol{v}\right)$, where $\boldsymbol{\mu}_\phi\left(\boldsymbol{h}, \boldsymbol{v}\right)$ and $\sigma_\phi^2\left(\boldsymbol{h}, \boldsymbol{v}\right)$ are neural networks. Hence sampling from Q_ϕ is straightforward, that is $\boldsymbol{z} = \boldsymbol{\mu}_\phi\left(\boldsymbol{h}, \boldsymbol{v}\right) + \sigma_\phi\left(\boldsymbol{h}, \boldsymbol{v}\right)\boldsymbol{\epsilon}$ for $\boldsymbol{\epsilon} \sim \mathcal{N}\left(\boldsymbol{0}, I\right)$.

Finally, both the the MDN in Eq. (5) and CVAE in Eq. (6) can be jointly trained using the following hybrid objective:

$$\mathcal{L}\left(\theta, \phi, \gamma\right) = \mathcal{L}_{\text{CVAE}} + \lambda \mathbb{E}_{z \sim Q_\phi}\left[\log P_{\gamma,\theta}\left(\boldsymbol{x} \mid \boldsymbol{v}, \boldsymbol{z}\right)\right] \tag{7}$$

for some $\lambda > 0$ to ensure the matching scale for both objective terms. Note that the density function $P_{\gamma,\theta}\left(\boldsymbol{x} \mid \boldsymbol{v}, \boldsymbol{z}\right)$ has two parameter sets (γ, θ), where θ is from the network underlying the generation model $P_\theta\left(\boldsymbol{h} \mid \boldsymbol{z}, \boldsymbol{v}\right)$ (as part of CVAE in Eq. (6)) and γ is from the network underlying the multimodal density function $P_\gamma\left(\boldsymbol{x} \mid \boldsymbol{v}, \bar{\boldsymbol{h}}\right)$ (as part of MDN in Eq.(5)). The two networks connect through the mean function $\bar{\boldsymbol{h}} = \boldsymbol{\mu}(\boldsymbol{v}, \boldsymbol{z})$. This enables backprop of gradient from $\boldsymbol{x}$ to $\boldsymbol{v}$.

Remarks We also experimented with Conditional GAN (CGAN) as an alternative for CVAE but CVAE is more competitive, suggesting that for CVAE is more suitable in explorative settings such as materials discovery. We also noted that there are more advanced versions based on the normalizing flow framework [12] where the Gaussian distribution is replaced by a more complex posterior. Our framework can be directly extended to use these more flexible approximation techniques but we leave this for future work and consider here the basic variational version.

2.4 Prediction

At test time, we often wish to sample specific design parameters from the conditional distribution $P(\boldsymbol{x} \mid \boldsymbol{v})$ given in Eq. (4). For each sample of the prior $\boldsymbol{z}$ we have a MDN of the form $P\left(\boldsymbol{x} \mid \boldsymbol{v}, \mu\left(\boldsymbol{v}, \boldsymbol{z}^{(i)}\right)\right)$. Since each MDN is a mixture model, there are also multiple ways to sample $\boldsymbol{x}$ for each $\boldsymbol{z}_j$. For evaluation, we use Gumbel sampling [10] for identify major modes of significant mass for each mixture model. The collection of all such modes constitutes our prediction set.

3 Results

We now demonstrate the effectiveness of the new methods CVAE+MDN in the domain of alloy discovery. In particular, we focus on Aluminum alloys, i.e., those mixed materials in which aluminum (Al) is the predominant metal. This represents one of the most important classes of alloys widely used in engineering and everyday products thanks to its light weight or corrosion resistance. We use Thermo-Calc[3] as simulator. The primary use of this software is to compute the *phase diagram* (representing target properties) for each alloy (representing design parameters). In our setting, a phase diagram is a distribution of phases of matters at each temperature. We use phase diagram as a proxy because materials designers can infer target properties from it.

3.1 Alloy composition

An Aluminum alloy composition consists of Aluminum as the main metal with highest proportion and the remaining metal elements used in this research are Cr, Cu, Mg, Ti, Zn, Zr, Mn, Si, and Ni. As a result, they have a mixture of metallic phases up to the melting point for all compounds. For example Aluminum alloy 2024 (see Fig. 3) has 7 metallic phases at temperature range 0-300 °C, then 6, 5, 3, and 2 metallic phases at temperature 350, 400-500, 550, and 600 °C respectively, and finally only 1 liquid phase from 650 °C onwards.

3.2 Description of input phase diagram

We give as input the elements and its fractions into the Thermo-Calc software and get it to simulate and generate distribution of the metallic phases across temperatures as output. The temperature is varied from 0 to 1500°C with step 50°C resulting in 31 temperature points. There are 49 unique phases across our dataset. This makes the phase diagram a matrix of size 49×31. Fig. 3 shows the phase diagram of the four example alloys mentioned above. We can see that only a small subset of metallic phases (compound solutions) are active at each temperature.

3.3 Datasets

We created two datasets for our experiments. The first dataset is of size 15,000 alloys from the following 30 known series of Aluminum alloys: 2014, 2018, 2024, 2025, 2218, 2219, 2618, 6053, 6061, 6063, 6066, 6070, 6082, 6101, 6151, 6201, 6351, 6463, 6951, 7001, 7005, 7020, 7034, 7039, 7068, 7075, 7076, 7175, 7178, and 7475. The dataset was generated as follows. First, for each base alloy, we varied its 9 auxiliary elements (Cr, Cu, Mg, Ti, Zn, Zr, Mn, Si, Ni) by a relative amount of $\pm 20\%$ and make sure that the total proportion of the 10 elements is 100%. Second, we fed these alloy compositions into the Thermo-Calc software and ran the physical simulation. The simulation pressure parameter was set to 1 atmosphere, and temperature was varied from 0 to 1500°C with step 50°C. Finally, the distribution of phases across 31 temperature points for each element composition was generated as output.

[3] http://www.Thermo-Calc.com

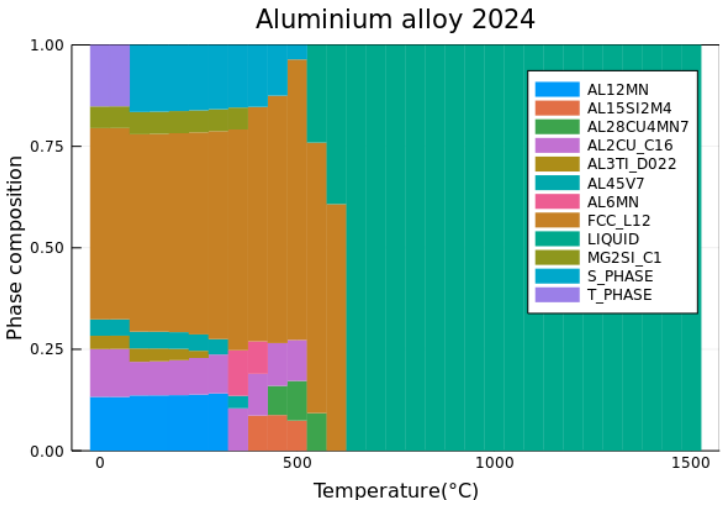


Figure 3: Phase diagram alloy 2024. The phases are square-root scaled then re-normalized for plotting purpose. Best viewed in color.

The second dataset is also of size 15,000 alloys and is created by searching via Thermo-Calc for a desired property in FCC phase by Bayesian optimization. In alloy design, the FCC phase proportion should be low ($< 95\%$) at low temperature and high ($> 98\%$) at high temperature. We investigated the FCC phases of known alloys at 200°C and 500°C. We found that these FCC phases have roughly a linear trend. Let y_{200} and y_{500} denote the FCC phases at 200°C and 500°C. We simply fit a regression line $y_{500} = ay_{200} + b$ to estimate $[a, b] = [0.45, 0.56]$.

We use Bayesian optimization and search in the domain of 9 auxiliary elements $E = \{\text{Cr, Cu, Mg, Ti, Zn, Zr, Mn, Si, Ni}\}$ for the phases close to this regression line. The Bayesian algorithm inputs the composition into Thermo-Calc and gets an objective as a function of the Thermo-Calc output. We carried out the search for one week period and yielded 120 search results. We randomly collect 1000 trajectory points satisfying the mentioned condition and vary their composition by $\pm 20\%$, each point 15 times, resulting in a dataset of size 15000 data points.

For our inverse problem, we used the phase diagram as input $\boldsymbol{y}$ and trained models to predict the element composition output $\boldsymbol{x}$. We use k-fold validation, that is, the dataset is randomly split into 5 folds, among which 4 folds are used for training and 1 fold is used for testing. The folds are alternated and the experiment is repeated 5 times. Mean and standard deviation of errors are reported.

3.4 Models setup

We use 7 different models:

1. *Baseline 1*: A Random Forests (RF). This model fits 30 randomized decision trees on random sub-samples, then uses averaging to avoid over-fitting. The max-features parameter is set to 100.
2. *Baseline 2*: A standard multi-layer perceptron (MLP) with three hidden layers of size 500, 100, and 50 respectively. The ReLU activation function is used for the hidden layers.
3. A mixture density network (MDN). This model has three sub-networks for the component means, variances, mixing weights respectively to make a mixture of Gaussians as output. These networks have the same hidden sizes as the MLP above.
4. The proposed conditional variational autoencoder MLP (CVAE-MLP). There are three components corresponding to the three conditional distributions (Eq. (3): 1) the recognition model $P(\boldsymbol{z}|\boldsymbol{v})$ to estimate the hidden posterior, this model has inside two sub-networks for the mean and standard deviation of the latent vector $\boldsymbol{z}$; 2) the generation network $P(\boldsymbol{h}|\boldsymbol{v}, \boldsymbol{z})$ to generate the unspecified property given the specified one and random noise; and 3) the prediction network $P(\boldsymbol{x}|\boldsymbol{v}, \boldsymbol{h})$ to predict the design parameters. These component networks have similar hidden sizes to the above MLP except that the second component has reverse order of layers. The latent size 30.

	Known-alloy dataset		BO-search dataset	
Method	**Relative (%)**	**Absolute (%)**	**Relative (%)**	**Absolute (%)**
RF	3.21 ± 0.02	0.00 ± 0.00	6.37 ± 2.13	0.01 ± 0.00
MLP	1.10 ± 0.03	0.00 ± 0.00	3.41 ± 1.48	0.01 ± 0.01
MDN	0.52 ± 0.00	0.00 ± 0.00	2.95 ± 1.32	0.00 ± 0.01

Table 1: Errors for completed phases input, averaged across 5 folds. For MDN, the mode with lowest error is reported.

5. The proposed CVAE-MDN. This model is similar to CVAE-MLP except the prediction network is MDN.
6. The alternative conditional generative adversarial network MLP (CGAN-MLP). This also has three components: 1) the generator $P(\boldsymbol{h}|\boldsymbol{v}, \boldsymbol{z})$; 2) the discriminator to decide whether $\boldsymbol{h}$ is real; and 3) the prediction network $P(\boldsymbol{x}|\boldsymbol{v}, \boldsymbol{h})$. The networks also have similar hidden sizes as above.
7. The CGAN-MDN. This model is similar to CGAN-MLP except the prediction network is MDN.

All neural network models were trained using ADAM optimizer with learning rate 0.001 until convergence with mini-batch size of 50.

3.5 Performance measures

Alloys vary in their composition, not just the proportion of elements, but also the existence of elements. On average the proportion of zeros of the alloy elements is about 20% in our dataset. That is, each alloy is composed of about 8 out of 10 elements. Given this fact, we reported two prediction errors: Relative Error for those elements which are nonzero and Absolute Error for those which are zero. Let $\boldsymbol{x}_i = (x_{i1}, x_{i2}, ..., x_{iM})$ be the true composition for instance i, and $\hat{\boldsymbol{x}}_i$ be the predicted composition. Then the relative error (for non-zero element $x_{ij} > 0$) and absolute error (for zero element $x_{ij} = 0$) are defined by:

$$r_{ij} = \frac{|\hat{x}_{ij} - x_{ij}|}{x_{ij}}; \qquad a_{ij} = \hat{x}_{ij} \tag{8}$$

The relative and absolute errors for the test set are computed as:

$$r = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{M_{i1}}\sum_{j=1}^{M} r_{ij}; \qquad a = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{M_{i0}}\sum_{j=1}^{M} a_{ij}$$

where M_{i0}, M_{i1} are the number of zeros and non-zeros elements of $\boldsymbol{x}_i$ respectively.

3.6 Prediction accuracy

We carried out several experimental settings to study the model ability in predicting alloy compositions. In the first experiment, we tried the prediction using the full phase matrix as input, see Fig. 3. In the following experiments, we compared different models and investigated whether they can generalize given an imprecise input.

Predicting element compositions Table 1 shows the error rate for this task using RF, MLP, and MDN. With the present of full phase matrices, predicted compositions has low error rates. For the known-alloy dataset RF can achieve 3.21% relative error, while MLP performs better at 1.1%. MDN has the lowest error, 0.5%. For the BO-search dataset RF, MLP and MDN achieve 6.37%, 3.41%, and 2.95% relative errors respectively. This supports the hypothesis that the output have multiple modes.

Predicting element compositions using partially known phases Table 2 compares different methods in predicting element compositions when the input phases are partially known, only 50% of the phases are presented to the model. Since the CVAE layer outputs a distribution instead of a single vector, we take 20 samples $\boldsymbol{z}$'s ($\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{0}, I)$) for each partial input $\boldsymbol{v}$ and output 20 fully

	Known-alloy dataset		BO-search dataset	
Method	**Relative (%)**	**Absolute (%)**	**Relative (%)**	**Absolute (%)**
RF	4.38 ± 0.01	0.00 ± 0.00	8.49 ± 1.34	0.01 ± 0.01
MLP	3.43 ± 0.07	0.00 ± 0.00	11.91 ± 2.54	0.03 ± 0.02
MDN	2.28 ± 0.22	0.00 ± 0.00	7.83 ± 1.11	0.01 ± 0.01
CVAE-MLP	2.50 ± 0.24 (a)	0.00 ± 0.00	7.42 ± 2.03 (e)	0.01 ± 0.01 (i)
CVAE-MDN	2.08 ± 0.12 (b)	0.00 ± 0.00	4.23 ± 0.67 (f)	0.00 ± 0.00 (j)
CGAN-MLP	3.18 ± 0.18 (c)	0.00 ± 0.00	8.39 ± 2.33 (g)	0.00 ± 0.00 (k)
CGAN-MDN	2.30 ± 0.18 (d)	0.00 ± 0.00	7.38 ± 0.70 (h)	0.00 ± 0.00 (l)

Table 2: Errors for partial phases input (50% phases missing). (a) ave: 3.41 ± 0.26, max: 4.45 ± 0.28; (b) ave: 2.50 ± 0.11, max: 3.34 ± 0.39; (c) ave: 3.33 ± 0.20, max: 3.5 ± 0.22 (d) ave: 2.58 ± 0.17, max: 4.52 ± 0.15; (e) ave: 8.62 ± 1.83, max: 10.71 ± 2.76; (f) ave:6.28 ± 1.34, max: 8.82 ± 3.52; (g) ave:10.45 ± 3.17, max: 12.85 ± 6.46; (h) ave: 9.68 ± 0.48, max: 11.39 ± 2.22; (i) ave: 0.01 ± 0.01, max: 0.01 ± 0.01; (j) ave: 0.0 ± 0.0, max: 0.01 ± 0.01; (k) ave: 0.02 ± 0.01, max: 0.05 ± 0.07; (l) ave: 0.02 ± 0.02, max: 0.02 ± 0.04. .

reconstructed $\bar{\boldsymbol{h}}$'s at the CVAE layer. Then these 20 reconstructions, combined with $\boldsymbol{v}$, are then passed to the MLP/MDN layer to predict 20 output $\bar{\boldsymbol{x}}$'s. The minimum, mean, and maximum errors for these $\bar{\boldsymbol{x}}$'s against the single target $\boldsymbol{x}$ are reported. The effect of reconstruction using CVAE is clearly demonstrated. For the first dataset the error of MLP drops by 0.93%, from 3.43% (without reconstruction) to 2.50% (with reconstruction). Likewise, the error of MDN drops from 2.28% (without reconstruction) to 2.08% (with reconstruction). Similarly, for the second dataset the error of MLP drops by 4.49%, from 11.91% to 7.42%. The error of MDN drops by 3.6%, from 7.83% to 4.23%. The CGAN-based models also reduce the errors for the basic MLP and MDN but not as good as the CVAE-based models.

We also experimented with different missing phase ratio for the MLP model and found that the error rate is still low (<10%) even at 70% missing rate. Only at 80% missing rate the error becomes significant (about 70%). This suggests that there is a small number of phases (mainly the compounds) that contains most of the element composition information.

Example of phase diagrams having mixture outputs Given an imprecise input (a partial designed phase diagram) there can be multiple alloy compositions satisfying this design. For an example, let the design phase be ($\text{fcc}_{200} = 0.915$, $\text{fcc}_{500} = 0.99$), then this is satisfied by three different alloys 7075, 7076, and 7175. The 7075 and 7175 have Cr proportion of 0.55 and no Mn element. while the 7076 has Mn=0.33 and no Cr. Also the 7075 has double the Ti proportion compared to the 7175. Similarly, for the design phase with ($\text{fcc}_{200} = 0.985$, $\text{fcc}_{500} = 0.998$) there are several 6000 series alloys satisfying it.

3.7 Verifying the error in Thermo-Calc

Although the prediction of design parameters have been fairly accurate with $2 - 3\%$ relative error, it is possible that the design may be at the edge of the plausible design region. Thus it is important to verify whether we can reconstruct the original specification given the prediction using the simulator itself. In this experiment, the predicted element compositions (20 output samples for one partial input phase diagram) for the partially known phases are fed back into Thermo-Calc for simulating the new phase diagrams. The phase diagrams, combined from the predicted property $\bar{\boldsymbol{h}}$ and the specified property $\boldsymbol{v}$, are then compared to the original phase diagram $\boldsymbol{y}$ to check whether the variational method has produced consistent elements for these phases. We expect the observed error to be small.

The absolute error for 1 phase diagram and 1 Thermo-Calc simulated diagram from the predicted alloy (using MLP) is: $\left\|\boldsymbol{h} - \bar{\boldsymbol{h}}\right\| = 0.03$ (each phase of the diagram is scaled to have the global range $[0, 1]$ across the dataset). The relative and absolute errors for observed phases between one partial input phase diagram and 20 simulated diagram from 20 predicted alloy compositions are 2.94% and 0.00% respectively (3,000 test examples). The relative error for phases are calculated similarly to the relative error for element compositions for nonzero phases in Section 3.5.

3.8 Computational efficiency and comparisons with the search-based methods

Recall that we can, in theory, perform extensive search for the best design parameters $\boldsymbol{x}$ for a given target properties $\boldsymbol{y}$ by running repeatedly the simulator to produce $\boldsymbol{y}$ from each $\boldsymbol{x}$. The time cost is the product of the time per simulator call, and the total number of calls. For this we compare our method with popular black-box search methods (i.e., genetic algorithm) for this inverse design problem. As a demonstration, we carried out the search for a random desired phase diagram, its true composition is {Cr=0.24, Mg=2.96, Ti=0.06, Zn=4.29, Mn=0.2, Si=0.12, Al=92.14}. For this, the trained CVAE-MDN gives immediately (in milliseconds) a set of 20 compositions, and accordingly 20 simulated phase diagrams by Thermo-Calc with a minimum of 0.7%, a mean of 2.0% , and a maximum of 3.36% relative error respectively.

For comparison we implemented random search, genetic algorithm (GA), and Bayesian optimization (BO). Random search fails to get any noticeable improvement after 1000 iterations. GA shows progress but after 1000 steps, the relative error is still far above the 50% mark. BO shows better performance than random search and GA. It took CVAE-MDN 1.825ms to find a solution without any search. Thermo-Calc took 80s for one run but the simulation-based search did not find any significant solution after 1000 runs (10+ hours, which is already 3-order of magnitude slower). We expect that it may takes days to achieve solutions of similar quality that is found with our learning-based solutions. The speed up by learning-based techniques (our methods) can be increased many folds *for free* since our methods can be run in batch (e.g., 100 examples) without a significant increase in time, thanks to the modern GPU architectures. For example, a batch of 30 alloys took only 2ms on the GPU in our experiment. The experiments were taken on the system with Intel Xeon E5-1650 v4 3.60 GHz CPU, Nvidia GeForce GTX 1080 Ti GPU, and 32GB RAM.

4 Related Work

Simulators offer a test-bed to perform a search or optimization to discover products with intended target properties [6, 17]. The availability of massive data sets collected from all experimental and simulation activities has led to the rise of data–intensive discovery using machine learning [3, 8, 11]. All these works have focused on harnessing data from laboratories and/or simulators to build re-usable machine learning models that can be queried much faster than simulators and therefore speed up the materials design process. To the best of our knowledge, our work is the first effort to completely eliminate the search/optimization involved in materials discovery process. Our inverse modeling can predict the intended design configuration instantly in a single step.

The problem of imprecise output specification has been discussed before in a limited context. For example, [7] assume a misspecification of output due to a small aleatoric uncertainty. However, none of the prior work allow the output specifications to be partially expressed or use imprecise specification to explore a class of products. Our approach to handle partial output specifications uses conditional variational auto-encoders, one of the current state-of-the-art models for unsupervised learning. Our model is related to recent deep nets for conditional density estimation (e.g., see [14, 16] for recent works and references therein). The main difference is we have incomplete input and multimodal output.

5 Conclusions

We have proposed a novel data–driven framework for designing new materials and products. Using the power of data mining, our framework has shifted the current design paradigm of searching for the target design to instant prediction. In day-to-day life of an experimental designer, this may easily bring a speed up of thousands (if not millions) of times. Although in this paper, we have demonstrated the applicability of our work mainly for designing alloys, our work is broadly applicable to most of the scientific design applications where it is possible to query simulators in offline mode or collect experimental design data in an ongoing basis. In the future, it would be useful to advance the scope of this research further by allowing qualitative specification of target properties (e.g. allowing specifications in term of preferences of component). Additionally, since the basic physical and chemical laws are often shared across different class of products, it may be interesting to use transfer learning across different product classes to achieve better prediction accuracy with smaller sized datasets.

References

[1] CM BISHOP. Mixture density networks. *Technical Report*, 1994.

[2] Rafiqul Gani and Efstratios N Pistikopoulos. Property modelling and simulation for product and process design. *Fluid Phase Equilibria*, 194:43–59, 2002.

[3] Surya R Kalidindi and Marc De Graef. Materials data science: current status and future outlook. *Annual Review of Materials Research*, 45:171–193, 2015.

[4] Larry Kaufman and Harold Bernstein. Computer calculation of phase diagrams. with special reference to refractory metals. 1970.

[5] Diederik P Kingma and Max Welling. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[6] XM Li and JJ Yu. Using orthogonal experimental design to optimize alloy composition. In *IOP Conference Series: Materials Science and Engineering*, volume 33, page 012065. IOP Publishing, 2012.

[7] Philipp Limbourg and Daniel E Salazar Aponte. An optimization algorithm for imprecise multi-objective problem functions. In *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, volume 1, pages 459–466. IEEE, 2005.

[8] Yue Liu, Tianlu Zhao, Wangwei Ju, and Siqi Shi. Materials discovery and design using machine learning. *Journal of Materiomics*, 3(3):159–177, 2017.

[9] Hans Leo Lukas, Suzana G Fries, Bo Sundman, et al. *Computational thermodynamics: the Calphad method*, volume 131. Cambridge university press Cambridge, 2007.

[10] Chris J Maddison, Daniel Tarlow, and Tom Minka. A* sampling. In *Advances in Neural Information Processing Systems*, pages 3086–3094, 2014.

[11] Paul Raccuglia, Katherine C Elbert, Philip DF Adler, Casey Falk, Malia B Wenny, Aurelio Mollo, Matthias Zeller, Sorelle A Friedler, Joshua Schrier, and Alexander J Norquist. Machine-learning-assisted materials discovery using failed experiments. *Nature*, 533(7601):73, 2016.

[12] Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*, 2015.

[13] Joseph L Schafer. Multiple imputation: a primer. *Statistical methods in medical research*, 8(1):3–15, 1999.

[14] Rui Shu, Hung H Bui, and Mohammad Ghavamzadeh. Bottleneck conditional density estimation. In *International Conference on Machine Learning*, pages 3164–3172, 2017.

[15] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In *Advances in Neural Information Processing Systems*, pages 3483–3491, 2015.

[16] Brian L Trippe and Richard E Turner. Conditional density estimation with bayesian normalising flows. *arXiv preprint arXiv:1802.04908*, 2018.

[17] Steven A Weissman and Neal G Anderson. Design of experiments (doe) and process optimization. a review of recent publications. *Organic Process Research & Development*, 19(11):1605–1633, 2014.