Humboldt-Universität zu Berlin
Mathematisch-Naturwissenschaftliche Fakultät
Institut für Informatik

Predicting Spin - Interaction Parameters of the Hamiltonian from Simulated INS Data of Single Crystals

Masterarbeit

zur Erlangung des akademischen Grades
Master of Science (M. Sc.)

eingereicht von: Eduard Gette
geboren am: 24.09.1992
geboren in: Pawlowka

Gutachter/innen: Prof. Dr. rer. nat. Lars Grunske
Dr. Jens-Uwe Hoffmann

eingereicht am: ..... verteidigt am: .....

Contents

1 Introduction

Elementary particles carry an intrinsic angular momentum, which is called *spin* due its loose resemblance to a quantity from classical mechanics associated with the motion of a rigid object around its center of mass [18, p. 165]. The magnetic properties of a material depend on the alignment of these spins, e.g. the spins of the electrons in the outermost region of the atom, called *valence electrons*.[47, p. 133]

Following the rules of thermodynamics, the ground state of the spins can be determined by finding the alignments that minimizes the equation $F = U - T * S$, where F is called the *free energy* of the total system, U is the internal energy, T is the temperature above absolute zero and S is the entropy.

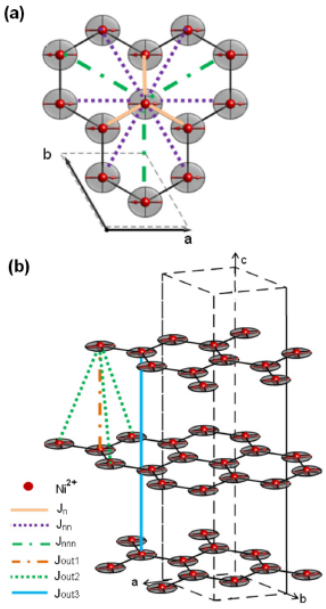

Figure 1: Crystal and magnetic structure of $BaNi_2V_2O_8$ within (a) a honeycomb plane and (b) a single unit cell. The red arrows represent spin directions. The magnetic exchange interactions are represented by the colored lines. [24]

When studying the magnetic properties of crystals under low temperatures, whose structure can be thought of as a 3-dimensional lattice of repeating elemental cells [8, p. 7] (figure 2) and is therefore highly ordered, the free energy term is dominated by the internal energy and therefore by the different types of interactions between the individual spins and their environment, e.g. external magnetic fields. (figure 1) Therefore theoretical calculations that are to be compared with experimental results, start from a model of this energy.

One such model is the Heisenberg model, also known in physics as the *Hamiltonian*, that describes the energy as a linear combination of the energies of different types of interactions. Due to the large number of particles involved, only a subset of all possible interactions can be considered to make the problem tractable. Which interactions should be included to properly model the system is usually decided in a phenomenological manner [47, p. 67].

Each type's contribution to the overall energy is given by a hyperparameter, that has to be estimated to model the given system. In practice these hyperparameters are estimated using an iterative approach analogous to the one shown in figure 3. For example in the case of this thesis the interaction structure is given by the Hamiltonian

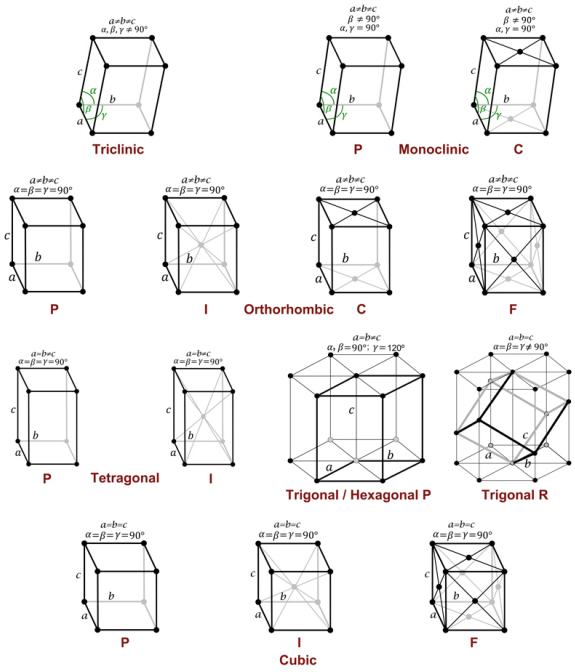

Figure 2: The 14 Bravais lattices and seven crystal classes possible in three-dimensional space. [29]

$$\mathcal{H} = \sum_{i>j} J_n \cdot S_i \cdot S_j + \sum_{i>j} J_{nn} \cdot S_i \cdot S_j + \sum_{i>j} J_{nnn} \cdot S_i \cdot S_j + \sum_{i>j} J_{out_1} \cdot S_i \cdot S_j$$
$$+ \sum_{i>j} D_{xy} \cdot S_i^{c^2} + \sum_{i>j} D_{inplane} \cdot S_i^{a^2}$$

, with J_n, J_{nn}, J_{nnn} and J_{out_1}, denoting the factor for the nearest, 2n-nearest, 3rd-nearest neighbor and inter-plane interactions respectively, while D_{xy} and $D_{inplane}$ denote the factors for the XY-plane and the in-plane anisotropy.

The *experimental data* is data resulting from **I**nelastic **N**eutron **S**cattering, short **INS**, experiments with the single crystal $BaNi_2V_2O_8$ and the theoretical spectra are obtained by sampling the hyperparameter space of the Hamiltonian and simulating the experimental outcome using the SpinW [44] software package.

INS experiments are a special kind of spectroscopy in which a monochromatic incident beam of neutrons is fired at a sample of the material of interest. Upon collision with the target material an incident neutron gets diffracted and some of its energy is absorbed by the sample, with the strength of both kinds of interactions being determined by the state of the sample's spins. Finally the *wave vector transfer* $\vec{Q}$, which represents the position in reciprocal space, and the *energy transfer* $\hbar\omega$ of the diffracted neutron are registered by a set of detectors behind the sample. Combining both of those properties in a graph provides a kind of magnetic profile for the material, called the ***D****ynamical*

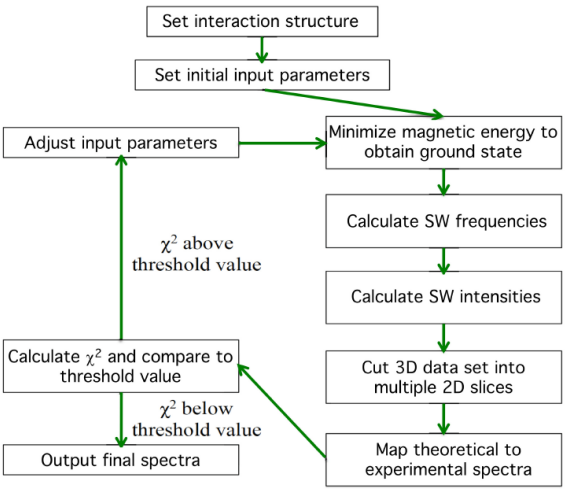

Figure 3: A flow diagram for the analysis of dynamical spectra. [13, p. (1-5)]

***S**tructure **F**actor* $S(\vec{Q},\omega)$ (**DSF**)[31, p.363-365].

However reliably fitting simulated to experimental data has proven to be difficult using classical approaches, e.g. minimizing χ^2, due to artifacts in experimental and simulated data causing the distance function to be noisy and flat around its minimum.[37]

Recent approaches to overcome those limitations in related fields of study attempt to leverage **M**achine **L**earning (**ML**) algorithms [10][37], that have proven to be successfully applicable to a wide range of problems in other domains.[49]

In this thesis the applicability of ML algorithms to directly solve the inverse transformation problem of regressing Hamiltonian parameters from simulated INS data of single crystal materials, is investigated using a previously constructed simulated data set of the single crystal $BaNi_2V_2O_8$.

To this end the thesis is structured in the following way.

First **Previous Work and Challenges** gives a short overview of current attempts to use ML to solve different problems in the field of crystallography to the best of the author's knowledge at the time of writing. Also the advantages and disadvantages of the two identified approach categories are outlined providing the reasoning for the choice of models and experiments in the consecutive chapters.

Next the chapter **Artificial Neural Networks and Variational Bayesian Methods** gives a short introduction to the most important concepts necessary to understand the algorithms and methods used in this work.

Chapter 4 introduces the data set, describing how the data was obtained as well as how feature engineering and pre-processing are performed in all experiments of the consecutive chapter.

The following chapter 5 summarizes a variety of different experiments performed on the data set and their methodology, with each one highlighting different modelling choices and how they affect different aspects of interest of the resulting algorithms, such as sharpness, calibration and accuracy of predictions or computational demand.

In chapter 6 all findings of the previous chapters are discussed and how they relate to the individual research questions investigated in this thesis.

Chapter **Conclusion and Future Work** provides a short overview of what has been achieved and highlights areas of interest for further research.

2 Previous Work and Addressed Challenges

Although at the time of writing of this thesis research on the applicability of ML for this specific problem is still sparse, two broad categories of approaches can be identified considering comparable problems in the wider context of determining properties of crystal materials from experimental data.

Either *(I) ML algorithms are being used as an intermediate step*, e.g. to narrow down the search space of downstream tried-and-proven methods, or *(II) a direct regression of the desired target variables from the inputs* is being performed.

2.1 ML-Assisted Approaches

An often debated issue of ML algorithms, especially those based on **A**rtificial **N**eural **N**etworks (**ANN**), is their black-box behavior. Gaining insights into the inner workings of such an algorithm to understand how it makes predictions and how those predictions are affected by changes in the input is still non-trivial and spurred an active field of research of its own. A natural approach in natural sciences, where fitting explainable models to experimental data is key, is therefore to solely restrict use of ML algorithms to areas where traditional approaches have proven unreliable or too time consuming.

For example in a very closely related problem A. M. Samarakoon et al. combined *traditional* iterative least-squares fitting to match simulation to experimental data with a **V**ariational **A**uto **E**ncoder [23] (**VAE**). The VAE was trained to create lower-dimensional embeddings of the data that are considered to contain only the most relevant information, removing unwanted noise and artefacts. These embeddings are then used in the fitting process to provide a more robust distance measure. Their algorithm suggests multiple sets of parameters by using a tolerance threshold, allowing for a quantitative measure of result uncertainty. [37]

Other more remotely related examples of ML-assisted efforts in the field of crystallography include:

1. Changwoo, Chen and Lee trained various simple ML models to create a pre-selection of scattering model classes that are likely to succeed in modelling the provided Small-Angle-Scattering (SAS) data [5]
2. Liu et al. used a **C**onvolutional **N**eural **N**etwork [26] (**CNN**) to suggest most likely space groups for a given atomic pair distribution function with a top-6 accuracy of 91.9 % [27]

Related challenges addressed by this thesis. Even though hybrid approaches are easier to interpret because they only apply black-box algorithms at non-critical steps to support tried-and-proven methods, they do not fully leverage the capabilities of artificial neural networks. For example the algorithm proposed by A. M. Samarakoon

et al. can only be used to predict discrete combinations of values sampled from the search space, while the optimal solution might be somewhere between those values. Although its granularity can be improved by increasing the number of iterations of the process, the time to find a solution scales linearly with the number of simulations.

For this reason in this thesis the following questions are addressed to alleviate the previously mentioned black-box problem of ANN-based approaches:

- How is *uncertainty quantification* performed in hybrid-approaches and what are the underlying assumptions/ limitations?
- How can some of those concepts be transferred to end-to-end approaches?
- Based on that, which steps effectively reduce uncertainty in end-to-end models?

2.2 End-to-End Approaches

Despite being hard to interpret ML algorithms have established themselves as state-of-the-art in a variety of fields, even outperforming humans in specific tasks such as playing the board game *Go*[38].

At the time of writing to the best of the author's knowledge no results on using ANNs to predict Hamiltonian parameters from single crystal INS data, real or simulated, in an end-to-end manner have been reported. Nevertheless several approaches to apply end-to-end ML algorithms to solve related problems have been developed, including:

1. Garcia-Cardona et al. use a shallow CNN to predict crystallographic symmetry classes and a random forest regressor to estimate unit cell parameters of crystal structures from simulated **S**cattering **L**ength **D**ensity (**SLD**) profiles [15]
2. Souza et al. employed different ML algorithms regularly used in computer vision tasks to solve the classification problem of whether an image contains a diffraction pattern or not [41]
3. Sullivan et al. formulate the problem of integrating Bragg peaks as a segmentation problem using a U-Net [36] to create Bragg peak masks [43]

Transferring the main concepts of the approaches in the aforementioned related topics of research to the task of regressing Hamiltonian parameters leads to the following questions that are investigated in later chapters:

Related challenges addressed by this thesis.

- Are ANN employed in an end-to-end manner capable of accurately predicting Hamiltonian parameters from simulated INS data?
- Which trade-offs have to be considered?
- How to determine an optimal set of hyperparameters?

3 Artificial Neural Networks and Variational Inference

3.1 Fundamentals of Artificial Neural Networks

Machine learning is a field of study concerned with the development and training of mathematical models and algorithms capable of automatically extracting meaningful patterns from large collections of data without being explicitly programmed. [2, p. 1-2]

One such class of algorithms are **A**rtificial **N**eural **N**etworks (**ANN**) (figure 4), often just called **N**eural **N**etworks (**NN**) or **M**ulti**l**ayer **P**erceptron **MLP**. [2, p. 229]

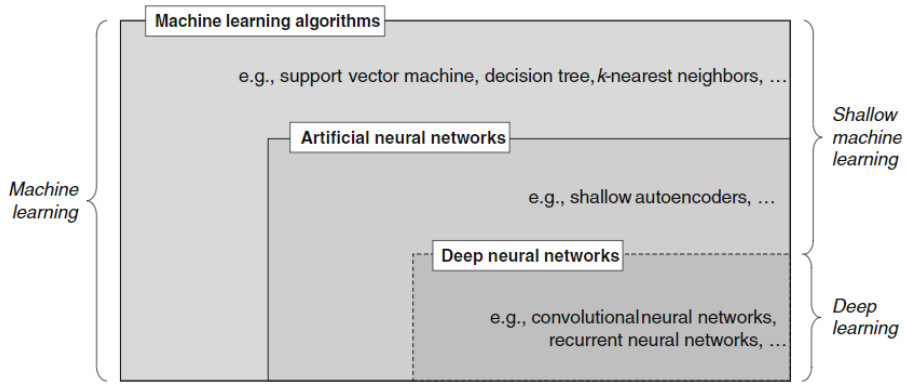

Figure 4: Venn diagram of machine learning concepts and classes. ([20], Fig. 1)

In the case of feedforward NN models the functional form is a series of functional transformations described by the equation:

$$z_j = h(\sum_{i=1}^{D} w_{ji}^{(l)} x_i + w_{j0}) \tag{3.1}$$

, where z_j with $j = 1...M$ denotes the so called *activation* of the j^{th} *unit*, l denotes the l^{th} step of the series, known as the *layer*, w_{ji} the *weight* parameter of the j^{th} unit for the i^{th} component of the *input* vector $\vec{x}$ and w_{j0} the corresponding *bias*. h is a differentiable, nonlinear function named *activation function*.[2, p.227-228]

Consecutive layers operate on the activations of the previous layer. All layers between the input and the last layer, the *output layer*, are called *hidden layers* and the total number of layers is commonly referred to as the *depth* of the NN. Besides learning a nonlinear mapping from their inputs to a useful representation, output layers have the additional purpose of shaping the output of the NN into a format suitable for the task the model is intended to perform.

Visually, feedforward networks are commonly depicted in the form of network graphs such as shown in figure 5.

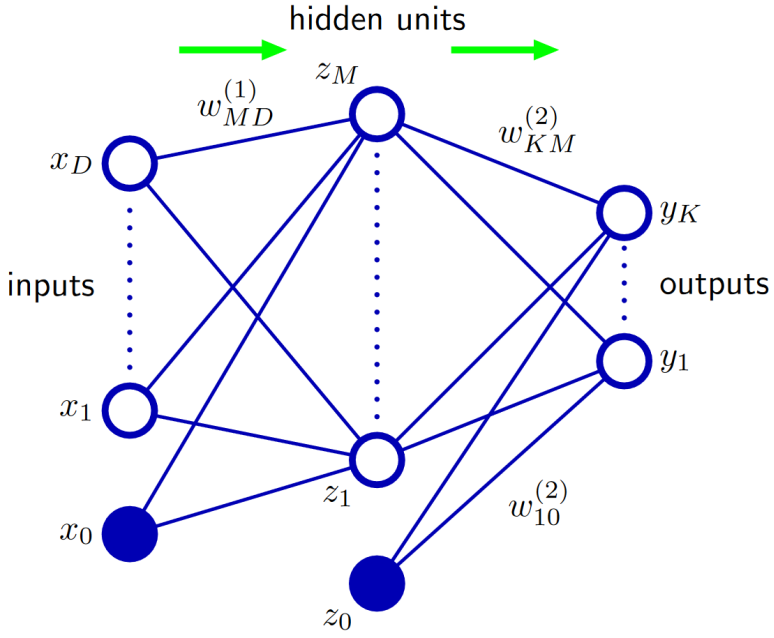

Figure 5: Network diagram for a feedforward neural network with two hidden **f**ully **c**onnected (**FC**) layers. Variables are depicted as nodes and weights as edges between those nodes. *Fully connected* describes the property that nodes in consecutive layers are connected to all nodes of the previous layer.([2], Fig. 5.1)

From a theoretical point of view feedforward NNs are universal approximators, meaning that they are capable of approximating any measurable function to any desired degree of accuracy given at least one hidden layer with a sufficiently large number of hidden units. A corollary of this result is that if a deterministic relationship between the input and the target exists, the quality of the approximation is solely dependent on an adequate choice of training procedure and hyperparameters, i.e. number of hidden units.[19]

The process of training a NN for a regression problem is best understood when analyzed from a probabilistic point of view. For the sake of simplicity the problem is reduced to regression of a single real valued target variable t, which is assumed to follow a Gaussian distribution with the mean being dependent on the network input x and a constant variance β^{-1}. Given these assumptions the conditional probability distribution to be approximated can be expressed as

$$p(t|x, w) = \mathcal{N}(t|y(x, w), \beta^{-1}) \tag{3.2}$$

Following a maximum likelihood approach, provided a data set of N independent, identically distributed observations X = $\{x_1, ..., x_N\}$ and a set of target values T = $\{t_1, ..., t_N\}$, the task of finding a set of weights w_{ML} that approximate the unknown distribution as closely as possible, is framed as an optimization problem, i.e. maximizing the likelihood function:

$$w_{ML} = \arg\max_w p(t|X, w, \beta) = \arg\max_w \prod_{n=1}^{N} p(t_n|x_n, w, \beta) \tag{3.3}$$

By convention ML tasks are formulated in terms of minimization of a so called *loss function*[17, p. 80] $\mathcal{L}$. For regression this loss function is the sum of squared errors, which can be derived from the previous equation by taking the negative logarithm of the likelihood function

$$\mathcal{L}(x, w) = \frac{\beta}{2}\sum_{n=1}^{N}(y(x_n, w) - t_n)^2 - \frac{N}{2}ln(\beta) + \frac{N}{2}ln(2\pi) \tag{3.4}$$

and finally dropping all constants, since they do not affect the optimization. The final optimization task is then given by

$$\arg\min_w \mathcal{L}(x, w) = \arg\min_w \sum_{n=1}^{N}(y(x_n, w) - t_n)^2 \tag{3.5}$$

After finding the optimal mean estimator $y(x, w_{ML})$, the estimated variance is given by

$$\sigma^2_{ML} = \beta^{-1}_{ML} = \frac{1}{N}(y(x_n, w_{ML}) - t_n)^2 \tag{3.6}$$

[2, p.233-234]

The most commonly used training algorithm to find w_{ML}, *gradient descent*, consists of two stages: (I) the feed-forward stage and (II) the backpropagation stage.

In the first stage the input data is passed in forward direction through all layers of the network and the error between the network's output and the targets is calculated. Then in the backpropagation stage the contribution of each weight to the total loss is calculated in the inverse direction, from the last to the first layer, by calculating the gradient of the loss function with respect to the models weights. Each weight's contribution is used to adapt the weight for the next iteration of the algorithm.

Computation of the exact gradient is very expensive when dealing with large data sets, because it requires to pass the entire data through the network for each iteration. As a result in practice it is more common to compute an unbiased estimate of the gradient instead, by taking the average gradient over small random subsets of the data, the so called *minibatches*[17, p. 290], cycling over the entire data set multiple times until convergence. Each cycle is referred to as an *epoch*. The simplest representative of this type of algorithm is ***S**tochastic **G**radient **D**escent* (**SGD**) (figure 6).

Algorithm 8.1 Stochastic gradient descent (SGD) update

Require: Learning rate schedule $\epsilon_1, \epsilon_2, \ldots$
Require: Initial parameter $\boldsymbol{\theta}$
  $k \leftarrow 1$
  while stopping criterion not met **do**
    Sample a minibatch of m examples from the training set $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ with corresponding targets $\boldsymbol{y}^{(i)}$.
    Compute gradient estimate: $\hat{\boldsymbol{g}} \leftarrow \frac{1}{m} \nabla_{\boldsymbol{\theta}} \sum_i L(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), \boldsymbol{y}^{(i)})$
    Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \epsilon_k \hat{\boldsymbol{g}}$
    $k \leftarrow k + 1$
  end while

Figure 6: Pseudo-code for the SGD algorithm[17, p. 291]

3.2 Convolutional Neural Networks

Convolutional **N**eural **Networks** (**CNN**) are special NN models from the class of deep neural networks (see figure 4) that employ the mathematical operation known as *convolution* or some variation of it, in place of general matrix multiplication in at least one of their layers. In most programming libraries, although still being called convolution by convention, the *cross-correlation* function, which for a two-dimensional input I and kernel K takes on the form

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i+m, j+n) K(m,n) \tag{3.7}$$

is implemented instead.[17, p. 326] Figure 7 provides a visualization for the operation specified by equation 3.7.

Convolutional layers possess four important properties that motivate their usage in deep learning, when dealing with data that has a known grid-like topology, like images or time series:

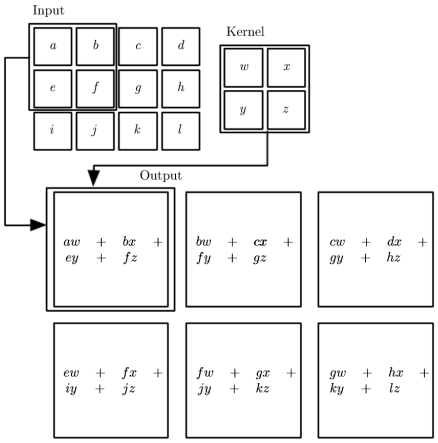

Figure 7: Example of 2-D cross-correlation. No padding is applied, so the output is restricted to areas where the kernel lies entirely within the image.([17], Fig. 9.1)

1. **sparse interactions**: Every output is only affected by a small subset of size k of the inputs, where k is the size of the convolution kernel which is usually several orders of magnitude smaller than the input's size m. This reduces the number of operations and required parameters to compute the output of one layer down from $(m * n)$ to $(k * n)$ per example. It is also noteworthy that although direct connections are sparse, deeper layers of CNNs, through the outputs of their previous layer, are indirectly connected to an increasing proportion of inputs, i.e. possess a larger *receptive field*. As a result the complexity of learned features of each layer increases in the direction of increasing depth.

2. **parameter sharing**: Each weight is not only used exactly once but rather for multiple functions in the same model, e.g. given an image input a single set of parameters is learned that is applied to every position in the image. Although not relevant regarding the runtime of forward propagation this further reduces the storage requirements of the model.

3. **equivariant representations**: The special kind of parameter sharing, as visualized by figure 7, gives the layers the property of equivariance to translation, meaning that a translation of the features in the input results in an equal translation of the activations of the output.

4. **input-size flexibility**: CNNs allow for a variety of mechanisms to handle inputs of varying size, e.g. through a combination of 1x1-convolutions and pooling operations, which is useful for many applications like image segmentation tasks.

[17, p.329-335]

3.3 Variational Inference and Bayesian Neural Networks

In contrast to the **m**aximum-**l**ikelihood **e**stimation (**MLE**) approach described by equation 3.3, which is employed by standard NN models and results in point-estimates for the optimal weights, in Bayesian inference the full posterior probability distribution of the weights is calculated. According to Bayes' theorem this posterior distribution can be computed as

$$p(\Omega|T) = \frac{p(T|\Omega) * p(\Omega)}{p(T)} \tag{3.8}$$

, where T is an observed variable dependent on a latent variable Ω, $p(T|\Omega)$ is the *likelihood*, *prior* $p(\Omega)$ is a distribution that captures prior information about Ω and $p(T)$ is the marginal probability distribution of observations known simply as *marginal* or *evidence*.

Unfortunately in many cases the computation of the marginal, given by

$$p(T) = \int_{w \in \Omega} p(T|w)p(w)dw \tag{3.9}$$

is either not available in closed form or takes exponential time to compute.[3]

Variational **I**nference (**VI**) is an approach to compute an analytical approximation of the intractable posterior distribution $p(T)$, that combines the computational speed advantage of MLE with the statistical benefits of the Bayesian approach.

To this end, first an approximation $q(\Omega)$ from some tractable family of distributions $\mathcal{Q}$ is selected. Next a set of parameters for this distribution is searched, that minimizes the distance, measured by some adequate distance measure, between the true posterior and the approximation such that $q^*(\Omega) \triangleq p(\Omega|T)$, where q^* denotes the best found estimation. As with many non-convex optimization problems there is a trade-off between accuracy of the estimation and time complexity of the computation, which can be controlled by relaxing constraints or using further approximations, e.g. of the optimization objective.[30, p. 731]

In VI a common choice for the distance measure is the inverse **K**ullback-**L**eibler (**KL**) divergence, also known as *I-projection* or *information projection*[30, p. 733]:

$$D_{KL}(q,p) = \sum_x q(x) ln(\frac{q(x)}{p(x)}) = \mathbf{E_{q(x)}}[ln(q(x)) - ln(p(x))] \tag{3.10}$$

Choosing the I-projection over the forward KL divergence stems from the observation that in practical applications the true posterior will often be multimodal, due to non-identifiability in the latent space or complex nonlinear dependence on the parameters.[2, p. 469] As shown in figure 8 minimization of $D_{KL}(p, q)$ would lead to an approximation that averages across all modes for common choices of $q(x)$, leading to poor predictive performance as the average of two good parameter values is usually not a good choice of parameters itself.[2, p. 469] Besides being more tractable to compute, the information projection is therefore the more sensible choice from a statistical point of view.[30, p. 733]

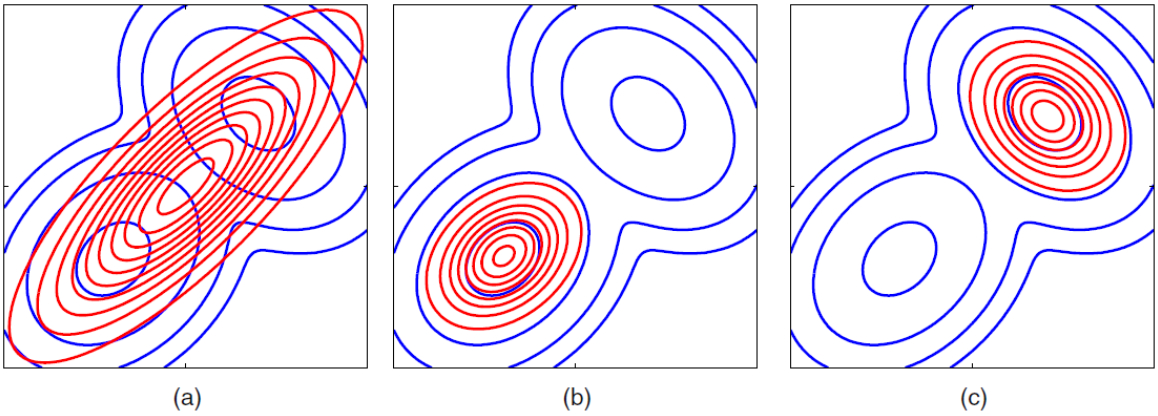

Figure 8: Comparison forward vs. inverse KL divergence. Blue contours show a bimodal distribution p(Z) to be approximated, red contours show the corresponding best-fit Gaussian distribution q(Z). (a) Best-fit q(Z) minimizing forward KL divergence $D_{KL}(p, q)$, (b) & (c) Two-possible best-fit distributions minimizing inverse KL divergence $D_{KL}(q, p)$ ([2], Fig. 10.3)

After defining the distance measure the optimization goal can be formulated as:

$$q^*(\Omega) = \arg \min_{q(\Omega) \in \mathcal{Q}} D_{KL}(q(\Omega), p(\Omega|T)) \tag{3.11}$$

Although optimizing this specific objective is still not tractable due to its dependence on computing the evidence, an alternative, tractable objective can be derived from equation 3.11, called the **E**vidence **L**ower **BO**und (**ELBO**):

$$\begin{aligned} \text{ELBO}(q(\Omega)) &= \mathbf{E}_{\mathbf{q}(\mathbf{\Omega})}[ln(p(\Omega, T))] - \mathbf{E}_{\mathbf{q}(\mathbf{\Omega})}[ln(q(\Omega))] \\ &= \mathbf{E}_{\mathbf{q}(\mathbf{\Omega})}[ln(p(T|\Omega))] - D_{KL}(q(\Omega), p(\Omega)) \end{aligned} \tag{3.12}$$

In the literature the ELBO is defined as the negative KL divergence plus $ln(p(T))$, which is a constant w.r.t. $q(\Omega)$, and therefore maximizing the ELBO is equivalent

to minimizing the KL divergence. Taking the negative ELBO finally yields the error function suitable for training a **B**ayesian **N**eural **N**etwork (**BNN**), known as the **V**ariational **F**ree **E**nergy (**VFE**):[4]

$$\text{VFE} = -\text{ELBO} = D_{KL}(q(\Omega), p(\Omega)) - \mathbf{E}_{\mathbf{q}(\mathbf{\Omega})}[ln(p(T|\Omega))] \tag{3.13}$$

The VFE is a sum of a data-dependent part, referred to as the *likelihood cost*, and the *complexity cost*, which depends on the prior over the network weights. An intuitive interpretation of how this cost function influences the training process is that it forces the training process to converge to a solution that captures the prominent patterns in the data as closely as possible, while imposing a form of regularization on the possible values of the weights through the prior in the complexity cost part.[4]

By definition, a neural network model is termed a Bayesian neural network, whenever a distribution is placed over the network parameters[16, p. 3] such as shown in figure 9. BNNs are therefore models that treat latent parameters as random variables for which a distribution can be estimated using Bayes' Theorem, conditioned on what is observable in the training data. Standard neural networks, in contrast, take on a *frequentist* perspective on the problem, where only the data is treated as random variables, while weights are assumed to have some true value that is just unknown.[16, p. 8]

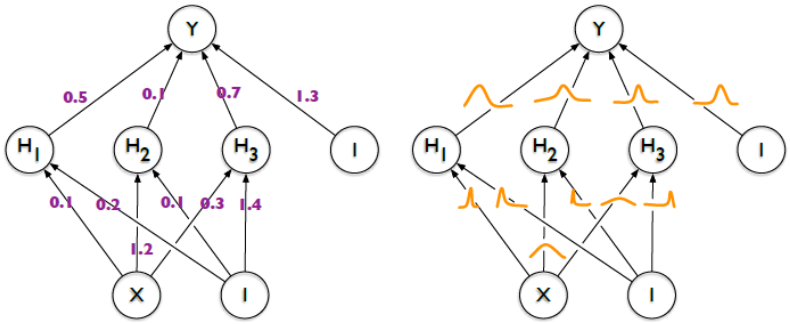

Figure 9: Comparison of how weights are modelled in standard NN vs. BNN. Left: each weight has a single fixed value. Right: each weight is assigned a distribution of possible values. ([4], Fig. 1)

4 The Data Set

4.1 Data Generation

Measurements of powder samples lack positional information in the reciprocal space, due to the random orientations of the individual crystals within the sample, whereas single crystal measurements do not suffer from this issue and therefore have the potential to improve prediction accuracy for this class of materials.

For this reason the given data set consists of samples obtained by running multiple simulations for a single crystal sample of the material $BaNi_2V_2O_8$, assuming a different configuration of interaction parameters for each run. As already mentioned in the introduction, the SpinW[44] software package was used for all simulations.

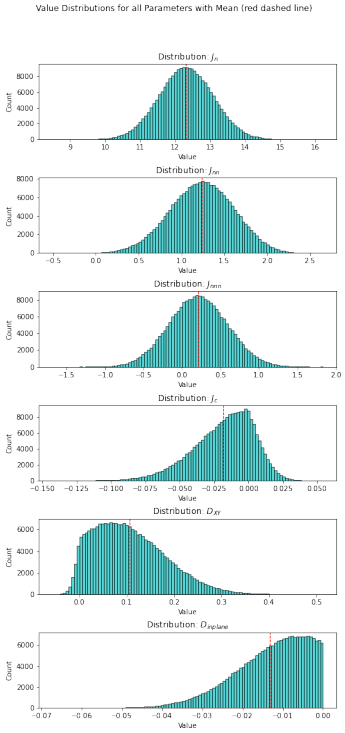

Figure 10: Distributions of ground truth labels for each interaction parameter.

Configurations were created randomly by sampling from six independent Gaussians, one for each parameter, at the beginning of each iteration, with the Gaussians being parameterized by mean and standard deviation as follows:

- J_n: $\mathcal{N}(12.3, 0.6)$
- J_{nn}: $\mathcal{N}(1.25, 0.15)$
- J_{nnn}: $\mathcal{N}(0.2, 0.15)$
- J_c: $\mathcal{N}(-0.00045, 0.001)$
- D_{XY}: $\mathcal{N}(0.0695, 0.01)$
- $D_{inplane}$: $\mathcal{N}(-0.0009, 0.0002)$

Besides discarding configurations for which the simulation could not be performed, no other filtering has been applied. The distributions of target values resulting from this data generation procedure are shown in figure 10.

4.2 Feature Engineering

Using the complete four-dimensional reciprocal space, resulting from simulations of single crystals, as input to train a NN model is computationally prohibitive.

Instead in order to reduce the data to a more manageable size and still capture most of the spatial information contained in the reciprocal space, different subsets of seven cuts along high symmetry directions are chosen as inputs to all algorithms. Specifications for all crystallographic directions are provided in figure 12.

4.3 Pre-Processing

All experiments conducted for this thesis use the same pre-processing scheme.

Experiments on *undistorted* data only apply min-max-normalization to the input-cuts followed by column-wise convolution with a Gaussian intensity-normalization-kernel (figure 11) to simulate the broader lines observable in real experiments due to limited instrument resolution. An example of the resulting inputs is given in figure 12.

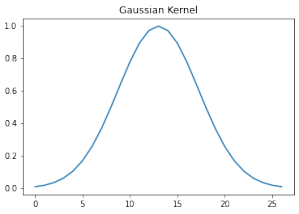

Figure 11: Gaussian Intensity-Normalization-Kernel.

Labels are always min-max-normalized to values between zero and one using the global maximum and minimum values for each parameter over the entire data set.

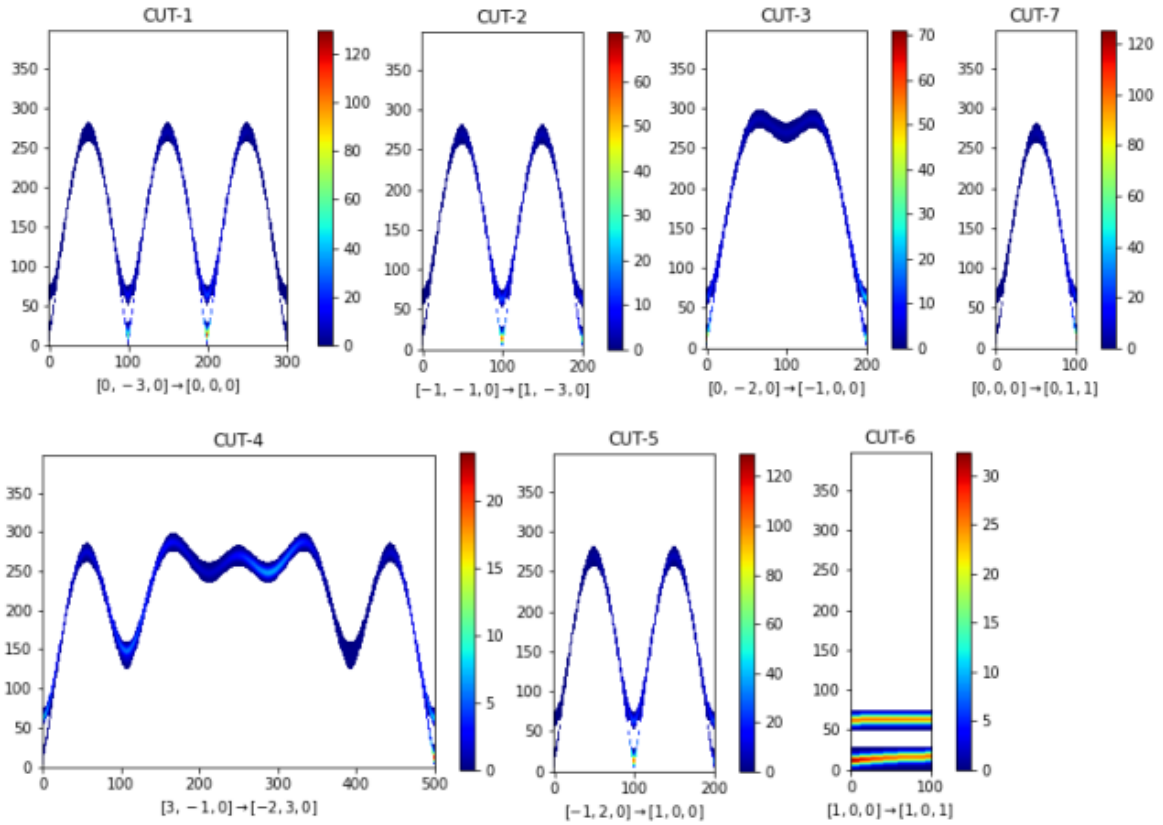

Figure 12: Overview: Dimensions and crystallographic directions of all seven cuts. Zero values are white for better visibility. The plotted example is the result of running the simulation with $J_n = 11.224705, J_{nn} = 1.405528, J_{nnn} = 0.135243, J_c = -0.007999, D_{XY} = 0.159600, D_{inplane} = -0.004701$ and applying column-wise convolution with the Gaussian kernel shown in figure 11. All parameter values are given in units of meV.

5 Experiments on Undistorted Simulation Data

5.1 Research Methods

5.1.1 Motivation

Introducing stochasticity into ANNs, i.e. through learning weight distributions using variational inference, provides multiple advantages when compared to frequentist models.

Revisiting the most important argument that NNs are statistical black-box models, whose decision process is mostly uninterpretable and solely justified through empirical means[16, p. 3], one key benefit is the ability to express uncertainty in these models and their predictions.[16, p. 23]

Another noteworthy argument is that Bayesian analysis provides meaningful theoretical foundations to reason about the effectiveness of some methods and algorithms currently used in many deep learning models, whose usage was previously justified only through empirical results.[16, p. 24] Dropout[42] is an example of such a widely used algorithm, for which it was shown by Kingma et al. that it can be considered a special case of the **S**tochastic **G**radient **V**ariational **B**ayes (**SGVB**)[23] method with local reparameterization.[22]

For the reasons given above the purpose of the following sections is to show through empirical means whether a BNN can be used to obtain reliable estimates of Hamiltonian parameters from the given data set. Due to the higher complexity incurred by using BNNs, which combine neural network models with stochastic modelling, the experiments are further designed to explore how the performance of the model is affected by different modelling choices and a lack of information in order to provide guidance on how to improve upon the models performance and which trade-offs to consider.

5.1.2 General Experiment Structure

Except for the first experiment, which introduces a first concrete implementation of a BNN model and therefore additionally serves as a baseline, all other experiments follow the same basic structure.

First the baseline model is tweaked accordingly to incorporate a single modelling choice to be investigated in the experiment. This tweak is done in a minimally invasive way, meaning that as much as possible of the baseline model is preserved. The reasoning behind starting from the baseline for every experiment is to isolate the effect of the tweak on the model's performance as much as possible.

Next the resulting model is trained from scratch on a training-split of the data set. During the training procedure the development of the loss and the root mean squared error on the training-split and another split of the data set, named validation-split, is monitored to be able to early detect issues in the training procedure such as over-fitting.

Finally the performance of the successfully trained model is evaluated on a separate test-split.

After every experiment a thorough performance comparison of the resulting model and the baseline is given to summarize the effect of the respective modelling tweak.

5.1.3 Evaluation

In order to be considered useful the BNN's predictions must be accurate, sharp and calibrated. Given a distribution of predictions for a single sample, the predictions are termed *accurate* if the mean of the distribution is close to the ground truth value and *sharp* if the distribution has low variance. Predictions are *calibrated* if the empirically observed confidence level is close to the expected confidence level. A model is termed *accurate*, *sharp* or *calibrated* if it outputs predictions with the respective property.

Estimate distributions are obtained by running the prediction step N_{PPS} times over the entire test split, with $N_{PPS} = 25$, resulting in 25 separate predictions per parameter per sample.

Getting the model's estimate $\hat{y}_i$ of the ground truth y_i for each sample x_i is done by taking the mean over all prediction vectors $\hat{y}_{i,j}$ of the i^{th} sample:

$$\hat{y}_i = \frac{1}{N_{PPS}} \sum_{j=1}^{N_{PPS}} \hat{y}_{i,j} \tag{5.1}$$

Confidence intervals are obtained similarly by calculating the respective upper and lower quantiles on a per component basis over all 25 predictions.

Following the definitions above the resulting models' performance in all experiments is measured by the following three metrics, each on a per target and over-all basis:

1. **M**ean **S**quared **E**rror (**MSE**):

$$\mathrm{MSE}^{\mathrm{k}}(\hat{Y}, Y) = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i) * (y_i - \hat{y}_i), k \in \{1, ..., K\} \tag{5.2}$$

$$\mathrm{MSE}(\hat{Y}, Y) = \frac{1}{K} \langle \mathrm{MSE}^{\mathrm{k}}(\hat{Y}, Y), \mathbf{1} \rangle \tag{5.3}$$

, where * denotes component-wise multiplication, Y the matrix of ground truth labels, $\hat{Y}$ the matrix of estimates for Y, N the total number of samples in the data set and K the total number of targets.

MSE measures the accuracy of predictions, i.e. how close predictions are to the ground truth value.

2. Calibration loss - cal (adapted from [25, p. 5, Eq. 9]):

$$\hat{p}_j^k = \frac{1}{N}||\{y_i^k | 1 \leq i \leq N \wedge LB(C_{i,j}^k) \leq y_i^k \leq UB(C_{i,j}^k)\}|| \tag{5.4}$$

$$\text{cal}^{\text{k}}(\hat{p}_1^k, ..., \hat{p}_m^k, p_1, ..., p_m) = \sum_{j=1}^{m} w_j (p_j - \hat{p}_j^k)^2 \tag{5.5}$$

$$\text{cal}(\hat{p}_1^1, ..., \hat{p}_m^K, p_1, ..., p_m) = \sum_{k=1}^{K} \text{cal}^{\text{k}}(\hat{p}_1^k, ..., \hat{p}_m^k, p_1, ..., p_m) \tag{5.6}$$

, where $C_{i,j}^k$ denotes the predicted j^{th} confidence interval for the k^{th} target of the i_{th} sample, $LB(C_{i,j}^k)$ its lower bound, $UB(C_{i,j}^k)$ its upper bound, y_i^k the k^{th} component of the ground truth vector for the i^{th} sample, $\hat{p}_j^k$ the empiric proportion of y_i^k that fall within the respective confidence interval and p_j the expected proportion.

Cal measures the reliability of the CI calculated from the predictions of the model, i.e. how close the empiric CIs are to the expected CIs. It is proportional to the area between the optimal line and the calibration curve in a reliability diagram.

3. Sharpness loss - sha (adapted from [25, p. 5, Eq. 10]):

$$\text{sha}^{\text{k}}(\hat{y}_{1,1}, ..., \hat{y}_{N,N_{PPS}}) = \frac{1}{N} \sum_{i=1}^{N} \text{var}^{\text{k}}(\hat{y}_{i,1}, ..., \hat{y}_{i,N_{PPS}}) \tag{5.7}$$

$$\text{sha}(\hat{y}_{1,1}, ..., \hat{y}_{N,N_{PPS}}) = \langle \text{sha}^{\text{k}}(\hat{y}_{1,1}, ..., \hat{y}_{N,N_{PPS}}), \mathbf{1} \rangle \tag{5.8}$$

, where var^{k} denotes a function which calculates the per-target variance over the k^{th} components of all provided $\hat{y}_{i,j}$.

Sha measures how strongly predictions of the model for the same sample deviate on average.

5.1.4 Hyperparameter Optimization

Neural network models possess a variety of hyperparameters, controlling every aspect from its functional or stochastic model to the speed of learning and the degree of generalization, rendering the concrete choice of hyperparameters one of the most important modelling decisions.

Besides their potentially high number, the effects of hyperparameters on the success of the training procedure are intertwined and must therefore be optimized simultaneously. In combination with the empiric observation that NNs tend to take a long time until they converge to a final set of weights, it is apparent that a brute-force random- or grid-search of the hyperparameter space will often either take too much time or will not find a high performing set of hyperparameters.

For the above mentioned reasons throughout all experiments KerasTuner's[33] implementation of the **B**ayesian **H**yperparameter **O**ptimization (**BHPO**) algorithm is used to determine the hyperparameters most relevant to the respective experiment. This implementation uses a **G**aussian **P**rocess (**GP**) prior with a kernel from the family of Matérn covariance functions with the common choice defaults $\nu = \frac{5}{2} = 2.5$ and $\theta = \mathbf{1}$. By using this specific choice of parameters, i.e. $\nu = p + \frac{1}{2}, p \in N^+$, the resulting Matérn 5/2 kernel can be simplified to a product of an exponential function and a polynomial of degree p as shown in the following equations, which conditions the GP to model the loss function as a smooth, two times differentiable function[35, p.84-85].

$$r^2(x, x') = \sum_{d=1}^{D} \frac{(x_d - x'_d)^2}{\theta_d^2} \tag{5.9}$$

$$K_{M52}(x, x') = \theta_0 * [1 + \sqrt{5}r(x, x') + \frac{5}{3}r^2(x, x')]e^{-\sqrt{5}r(x,x')} \tag{5.10}$$

The **L**ower **C**onfidence **B**ound (**LCB**) of the GP

$$\text{LCB}_{GP}(x, \kappa) = \mu(x) - \kappa * \sigma(x), \quad here \quad \kappa = 2.6 \tag{5.11}$$

is used as an acquisition function, with μ and σ being the predicted mean and the marginal standard deviation function of the fitted GP respectively, while κ is a hyperparameter that controls the exploitation-exploration trade-off.[39]

Intuitively the GP constructs a probabilistic model based on previous observations of the training process $\{x_n, y_n\}_{n=1}^{N}$, where x_n is a vector of previously tried hyperparameter values and y_n is e.g. the realized validation loss of the trained model.

Mean function μ encodes the GP's current estimate of the loss function as a function of the hyperparameters, while the marginal standard deviation function σ encodes the GP's uncertainty about the value of the function at each point in the hyperparameter space.

Acquisition function LCB_{GP} determines at each iteration of the BHPO algorithm which point of the hyperparameter space will be sampled for the next training procedure by

assigning a score to each point from a set of candidate points. The candidate point with the lowest score will be used for the next training procedure. If $\kappa = 0$ the candidate point that minimizes the GP's current estimate of the loss function will be selected, whereas large values of κ will entice the algorithm to select candidate points in areas where the GP's uncertainty about its estimate of the loss function is high.

5.2 Experiment 1: Baseline Bayesian CNN

5.2.1 Goal of the Experiment

The outcome of this experiment provides a first trained model, that serves as a baseline to evaluate all future experiments against.

5.2.2 Architecture / Functional Model

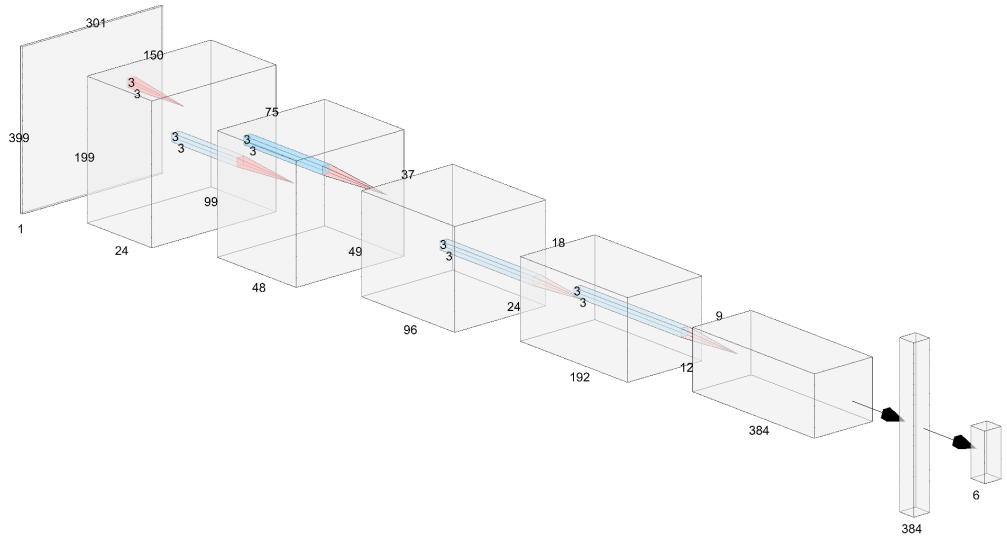

Figure 13: Architecture of the baseline bayesian CNN. The model consists of five convolution blocks followed by a channel-wise global average pooling layer. Each block applies (3x3)-convolution with (1,1)-strides and *same*-padding to the previous layer's output followed by a (2x2)-max pooling operation with (2, 2)-strides. The number of convolution kernels doubles after each block starting at 24 kernels. The output layer is a FC layer with six units applied to the flattened output of the previous layer. In total the model has approx. 1.7 mio. trainable parameters.

From a theoretical point of view a FC-NN model is capable of learning the concept of topology, that is hard-coded into CNNs, from the data, since CNNs can be seen as similar to a special subclass of FC networks with an infinitely strong prior that forces weights outside of the support of the kernel to zero and the weights for one hidden unit to be identical to the weights of its neighbor but shifted in space.[17, p. 341]

Another important observation is that for the given data set the exact spatial information is of great importance because both, the presence and the absence of diffracted neutrons

in every position in space, encode relevant information about the material.

Combined the previous two aspects suggest that in theory a FC-NN architecture is the best choice to model the data at hand, due to its capacity to extract the information encoded in the spatial distribution represented by the image data in a more general way than CNNs, which are limited by the in-built assumptions.

In practice however, the use of FC-NN architectures on image data has often proven to be impracticable due to the high computational demand and the intricacies of training models with a large number of parameters. Besides that, CNNs have been used in research to successfully solve a variety of problems due to the properties outlined in section 3.2, which allow them to learn to extract useful features from data with a grid-like topology, i.e. images, more efficiently than FC models.

As both, the high resolution of the input data and the additional parameters required for modelling uncertainty in BNNs, worsen the aforementioned issue of practicality of FC NN models by further increasing the computational demand, for the purpose of this thesis the architecture of the baseline model was chosen to be a CNN-architecture as shown in figure 13.

All layers, except the output layer, apply a variant of the Swish activation function[34] (figure 14) with the value of the β parameter being set to 1, , which is known as Swish-1 or **Si**gmoid-Weighted **L**inear **U**nit (**SiLU**)[11], on their outputs.

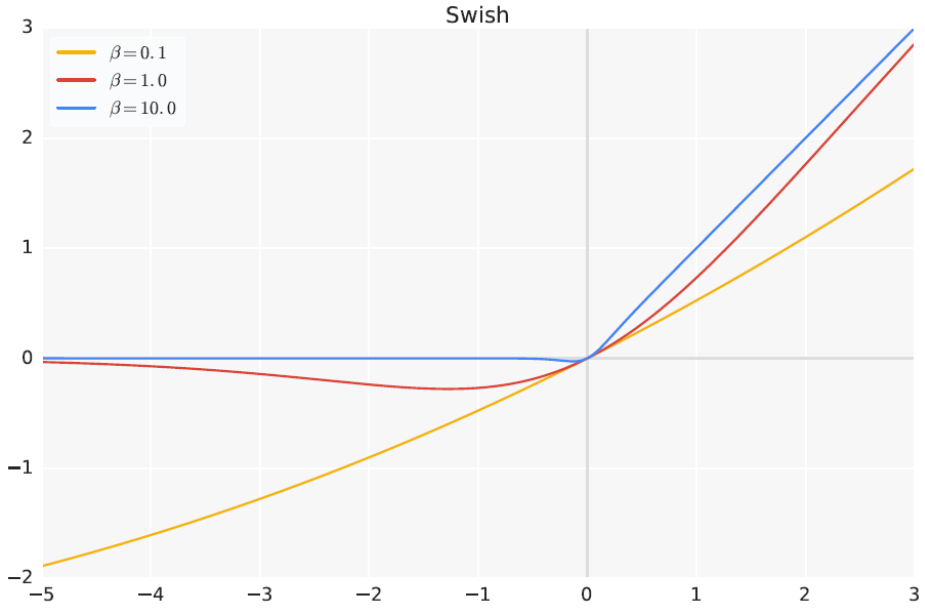

Figure 14: Swish activation function.([34], Fig. 4) The β parameter controls the degree to which the Swish activation function interpolates between the linear function $\text{linear}(x) = \frac{x}{2}$ and the **Re**ctified **L**inear **U**nit (**ReLU**) function $\text{ReLU}(x) = \max(0, x)$.[34]

Swish is defined as

$$\text{Swish}(x) = x * \sigma(\beta * x) \tag{5.12}$$

with σ being the sigmoid function $\sigma(z) = (1 + e^{-z})^{-1}$.

Usage of this specific activation function was motivated by the experiments conducted by P. Ramachandran et al.[34], which have shown for a variety of common activation functions, that the performance of multiple state-of-the-art models can be improved by simply replacing the previously used activation function by Swish without any further architectural optimizations. While β can be treated as an another tainable parameter in the network, setting $\beta = 1$ was enough to improve the performance in the experiments conducted by P. Ramachandran et al.[34] and S. Elfwing et al.[11] and is therefore a natural choice for the baseline model.

Choosing the number of units in the first convolution block and the total number of convolution blocks in the model was done using the previously introduced BHPO algorithm.

Regarding the implementation of the models and the training procedures the following Python libraries have been used in all experiments:

1. TensorFlow[28]
2. TensorFlow Probability[9]
3. Keras[6]

5.2.3 Stochastic Model

As already mentioned in section 3.3 the variational approximate distribution is to be selected from a family of tractable distributions.

To this end in **M**ean **F**ield VI (**MFVI**) the assumption is made that the true posterior can be approximated using a variational distribution that can be expressed as a product of distributions, one for each group of independent latent variables, i.e.

$$q(w_1, ..., w_M) = \prod_{i=1}^{R} q_i(w_{G_i}) \tag{5.13}$$

, where M denotes the number of weights, R the number of groups and w_{G_i} the set of weights that belong to the i^{th} group. Under this assumption the VFE can be computed efficiently if the prior $p(\Omega)$ and each q_i in equation 5.13 is set to be a Gaussian

distribution, since the second part - the KL divergence between weights' posterior and their prior - can be calculated in closed-form.[7]

Gaussian mean-field approximations, although often criticized for not being able to capture the true beliefs about the weights' distribution due to strong underlying assumptions, are still standard for modern Bayesian neural network inference[14] and therefore lend themselves as sensible choices for the baseline model.

Furthermore Farquhar et al. have presented evidence that as networks become deeper the performance gap between mean-field and structured-covariance approximate posteriors shrinks, indicating that for deep models **G**aussian MFVI (**GMFVI**) is capable of reasonably approximating the true posterior.[12]

Lastly as will be discussed in the following part, GMFVI also possesses other desirable properties that allow for a more stable training procedure.

All of the aforementioned arguments lead to following choice of stochastic model for the baseline model which corresponds to the default values of the respective layers' implementation in the TensorFlow Probability library:

1. Prior: $\forall \quad 1 \leq i \ \leq M \quad w_i \sim \mathcal{N}(0, 1)$
2. Initialization Posterior: $\forall \quad 1 \leq i \ \leq M \quad w_i \sim \mathcal{N}(0, 0.05^2)$

5.2.4 Training

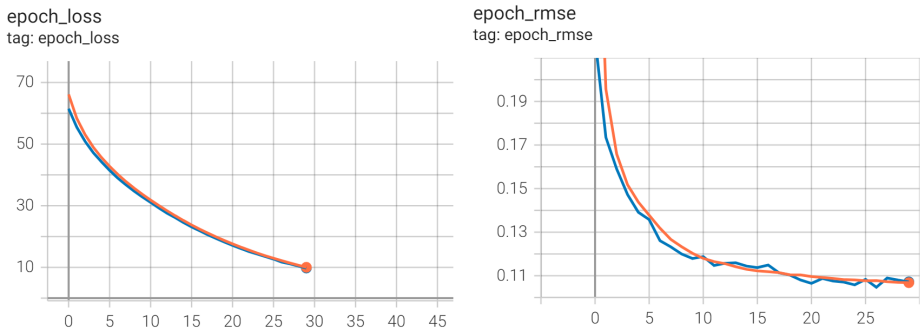

Figure 15: Overview: Progress of the training procedure. Left: VFE loss per epoch. Right: **R**oot **M**ean **S**quared **E**rror (**RMSE**) per epoch. Orange curve represents the corresponding quantity on the training split, blue on the validation split.

Training the baseline model was done for 30 epochs (figure 15) on a training data set of 32.000 samples of CUT-1 with a batch size of 16, using the *Adam*[21] optimizer with a learning rate of 0.00024974, found using BHPO, $\epsilon = 1e - 7$, $\beta_1 = 0.9$ and $\beta_2 = 0.999$. Validation and testing was done on separate splits of the data, each containing 3.200 samples.

Unless stated otherwise all experiments in section 5 are conducted on the same training-validation-test splits.

As previously stated, each weight is considered to be independent of the others, following its own Gaussian distribution, which is symmetric around zero. These two properties of the stochastic model are of special interest for the training process because combined they allow for efficient sampling of pseudo-independent weight perturbations for each sample within a mini-batch using the *Flipout* algorithm. Flipout achieves the ideal linear decay of $\frac{1}{B}$, with batch size B, of the stochastic gradients' variance, leading to significant reductions of training time.[45]

5.2.5 Evaluation

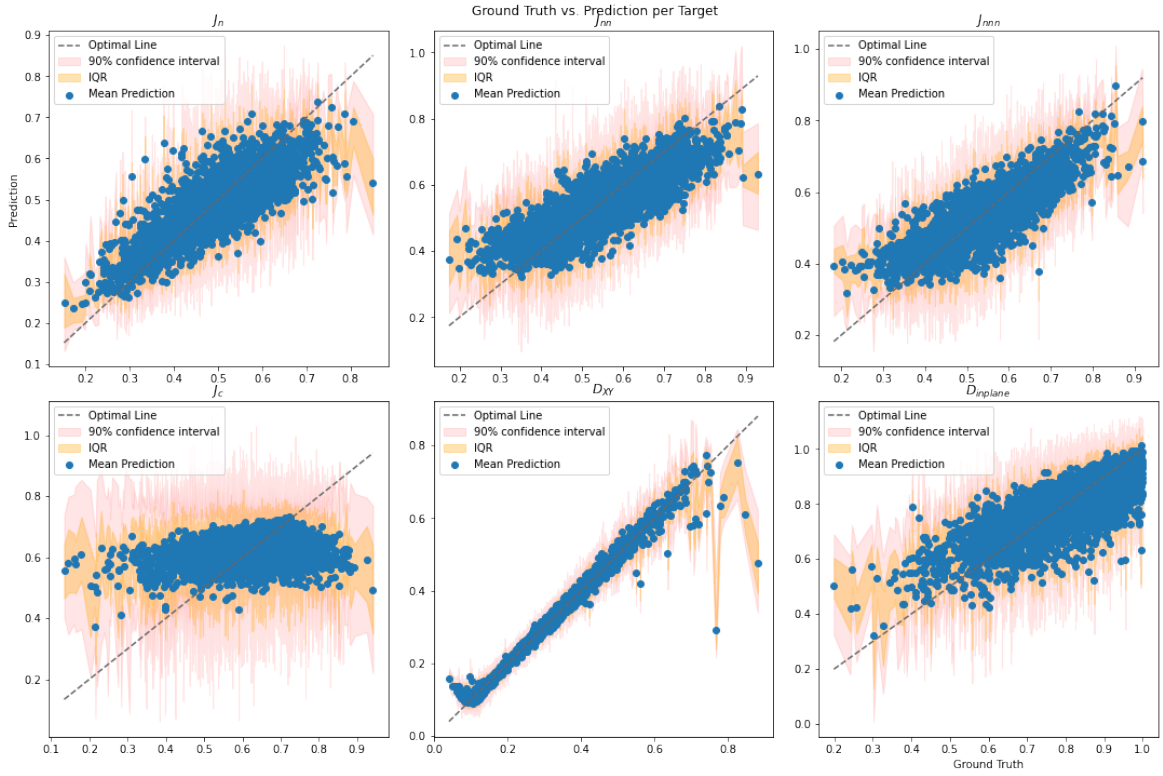

Figure 16: Ground Truth vs. Mean Prediction of the Baseline Model sorted in ascending order of ground truth y_i. The dashed gray line illustrates a perfect fit, $\hat{y}_i = y_i$. **IQR** is the **I**nter**q**uartile **Range**, which is the range between Q3 and Q1.

Analyzing figure 16 allows for several interesting observations.

First of all it is apparent that not all targets could be modelled equally well. D_{XY} for example has a very thin point cloud along the optimal line and narrow **C**onfidence **I**ntervals (**CI**) indicating that the model is capable of accurately predicting this interaction parameter with high confidence. On the other hand, the model seems

to have learned to always predict the mean of J_c, which is a poor estimate of the ground truth for most samples. Given that the baseline model was trained solely on samples of CUT-1 and that the confidence intervals for J_c are very broad, the most likely explanation for the poor performance on this target is that CUT-1, which only contains a small fraction of information about the reciprocal space, lacks information about the inter-honeycomb-plane interactions of the spins.

Inspecting the MSE and Sha row of table 1 quantitatively confirms the above observation, since both rows show different values for each interaction parameter with a maximum value at the J_c column and minimum value at the D_{XY} column.

Second, the model's predictions reflect that in the simulated data set some areas of the interaction-parameter space have been under-sampled, leading to deteriorated performance of the network in those areas. D_{XY} and $D_{inplane}$ are especially suited to support this claim, since they both have narrow tails on opposite sides of their respective distributions as shown in figure 10. While the low number of samples with values in the upper range of D_{XY} makes it harder for the BNN to accurately predict values in the normalized range above ~ 0.7, the same is true in the opposite direction, i.e. the lower end of the normalized value range, for values of $D_{inplane}$ below ~ 0.4.

Metric	Per Target Values						Over-all
	J_n	J_{nn}	J_{nnn}	J_c	D_{xy}	$D_{inplane}$	
MSE	0.00328	0.00475	0.00346	0.01182	0.00035	0.00711	0.00513
Cal	0.00249	0.00130	0.00419	0.00076	0.05040	0.00021	0.05934
Sha	0.00458	0.00688	0.00547	0.01277	0.00063	0.00873	0.03906

Table 1: Performance Summary of the Baseline Model. All values are rounded to five decimals.

From Table 1's second row it can be inferred that the CIs calculated from the baseline model's predictions are also not equally reliable across targets and that there should be intervals for which the model is either over- or under-confident.

Inspecting the per-target calibration curves (figure 18) shows that the model is almost perfectly calibrated for J_c and $D_{inplane}$, well calibrated for J_n, J_{nn} and J_{nnn} with a slight tendency towards being under-confident and highly under-confident for D_{XY}.

5.2.6 Influence of Training-Split Size on Performance

Besides the number of trainable parameters of the model, which is determined by BHPO after the training data is fully specified, the amount of required resources for training depends on the size of the individual samples and the total number of samples in the training-split of the data set. As previously stated in section 4, the size of the training inputs has been reduced by only providing cuts through the much larger reciprocal space to the network during training for the sake of feasibility. Further reduction of the

available information could severely limit the capability of the model to extract useful patterns from the data, leading to the choice of keeping the full resolution of the cuts.

Therefore to strike a balance between required training time and performance of the resulting model, the number of samples in the training-split is set to 32.000 samples. In order to show that this is a reasonable choice leading to meaningful predictions, figure 17 summarizes the effect of training-split size on the over-all performance of the baseline model trained from scratch two more times, using 10.000 and 20.000 samples respectively. While the effect on calibration is inconclusive, which is to be expected as it is the result of misspecification of the stochastic model and approximate inference, figure 17 shows that increasing the number of samples provides diminishing returns w.r.t. sharpness and accuracy and that the chosen cut-off value is made in accordance with the elbow method.

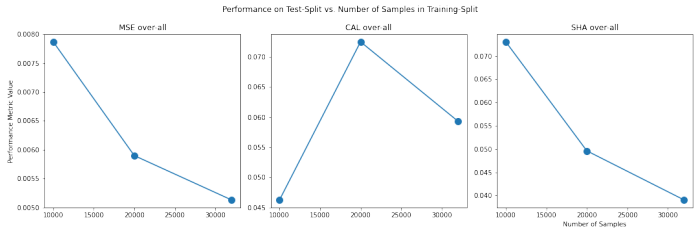

Figure 17: Summary: Performance vs. Size of the Training-Split of the Data Set.

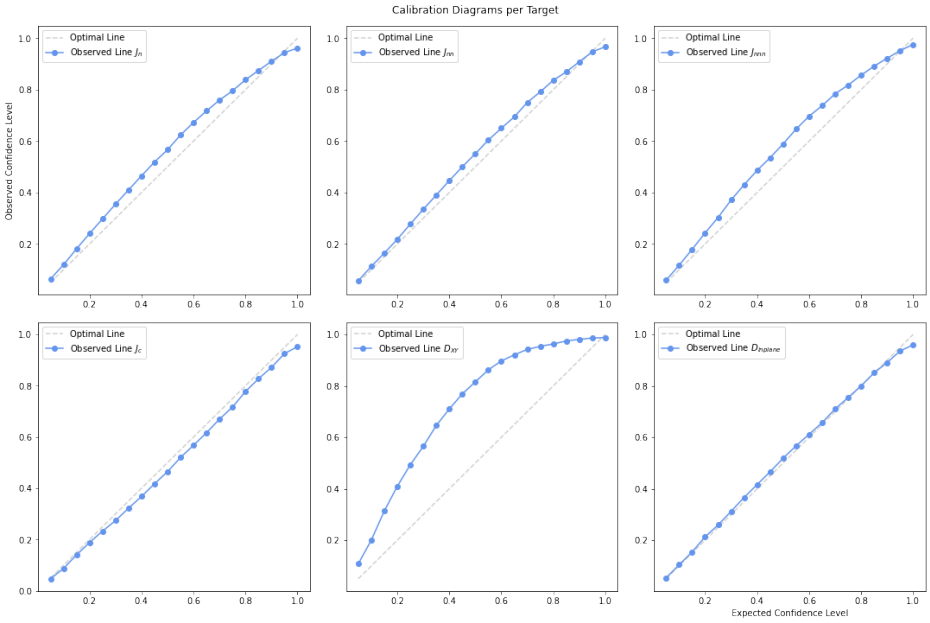

Figure 18: Per-target calibration curves of the baseline model. The abscissa shows the expected proportion of ground truth values that fall within the respective CI over the test split, while the ordinate shows the observed proportion over the test split. Values above and below the optimal line show that that the model's predictions are under-confident or over-confident respectively for the values in the corresponding CI's range.

5.3 Experiment 2: Influence of the Cold Posterior Effect

Although in theory the Bayesian posterior should give the optimal predictive performance, assuming the stochastic model is correctly specified[14], it is commonly observed in practice that the performance of BNNs can be improved significantly by artificially sharpening the Bayesian posterior.[46]

A common implementation to achieve this effect in BNNs, trained using GMFVI, is to introduce a hyperparameter, usually referred to as *temperature* $T \in [0, 1]$, which should not be confused with the temperature of the sample in an INS experiment, to the VFE loss function and optionally to the Gaussian prior of the weights, such that

$$\mathrm{VFE_T} = -\mathrm{ELBO_T} = T * D_{KL}(q(\Omega), p(\Omega)_T) - \mathbf{E_{q(\Omega)}}[ln(p(T|\Omega))] \tag{5.14}$$

$$p(\Omega)_T = \mathcal{N}(\mu, T * \sigma^2) \tag{5.15}$$

Choosing $T = 1$ leads to the original Bayesian posterior, while choosing $T < 1$ down-weights the complexity cost during training and reduces the prior's variance. Only down-weighting the complexity cost part of the loss function allows the network to over-count the importance of the data relative to the imposed regularization, while the down-scaled variance of the prior $T * \sigma$ allows for learning of narrower weight distributions that behave more like the weights in non-probabilistic neural networks. If only equation 5.3 holds the resulting posterior is called *tempered*, while applying T to both - VFE and prior - results in a *cold* posterior.[1]

The deviation of the empirical results from theory has motivated researchers to investigate the origins of this so-called **C**old **P**osterior **E**ffect (**CPE**). At the time of writing it appears that the CPE can arise from a variety of sources, including bad priors, data curation and data augmentation, and therefore in practice no simple fix seems to exist.[32]

5.3.1 Goal of the Experiment

In this experiment it is tested whether the CPE can be used to improve the predictive performance of the baseline model and how the sharpness and the calibration are affected by cold and tempered posteriors.

5.3.2 Training

Leaving all other aspects of the training procedure equal, the baseline BNN was trained from scratch twice as shown in figure 19, with the only modification being the aforementioned adaptations given by equations 5.3 and 5.3.

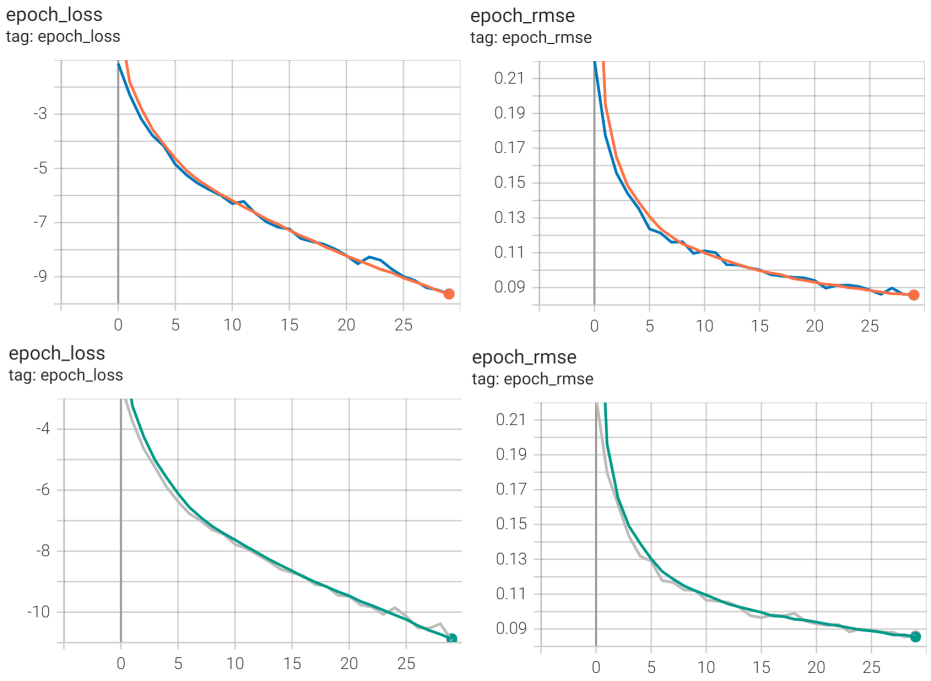

Figure 19: Overview: Progress of the training procedure. Left: VFE loss per epoch. Right: RMSE per epoch. Orange/Turquoise curve represents the corresponding quantity on the training split, blue/gray on the validation split. Top: Tempered Posterior, Bottom: Cold Posterior.

For the tempered posterior experiment an optimal temperature $T^*_{tempered} = 0.02682$ was found by the BHPO algorithm, while the optimal temperature for the cold posterior experiment was slightly lower with a value of $T^*_{cold} = 0.01681$.

5.3.3 Evaluation

Comparison of the over-all MSE values of table 2 shows that both modified posteriors lead to increased over-all performance on the test split over the baseline, with the tempered posterior model being slightly more accurate than the cold posterior model. Taking a closer look at the MSE per target values reveals that this performance gain is mainly achieved through more accurate predictions on the nearest-neighbor interaction parameters J_n, J_{nn} and J_{nnn}, as visualized in figure 21.

Regarding the calibration, both modified posteriors lead to higher Cal scores meaning that sharpening the posterior lead to deterioration of the baseline model's calibration.

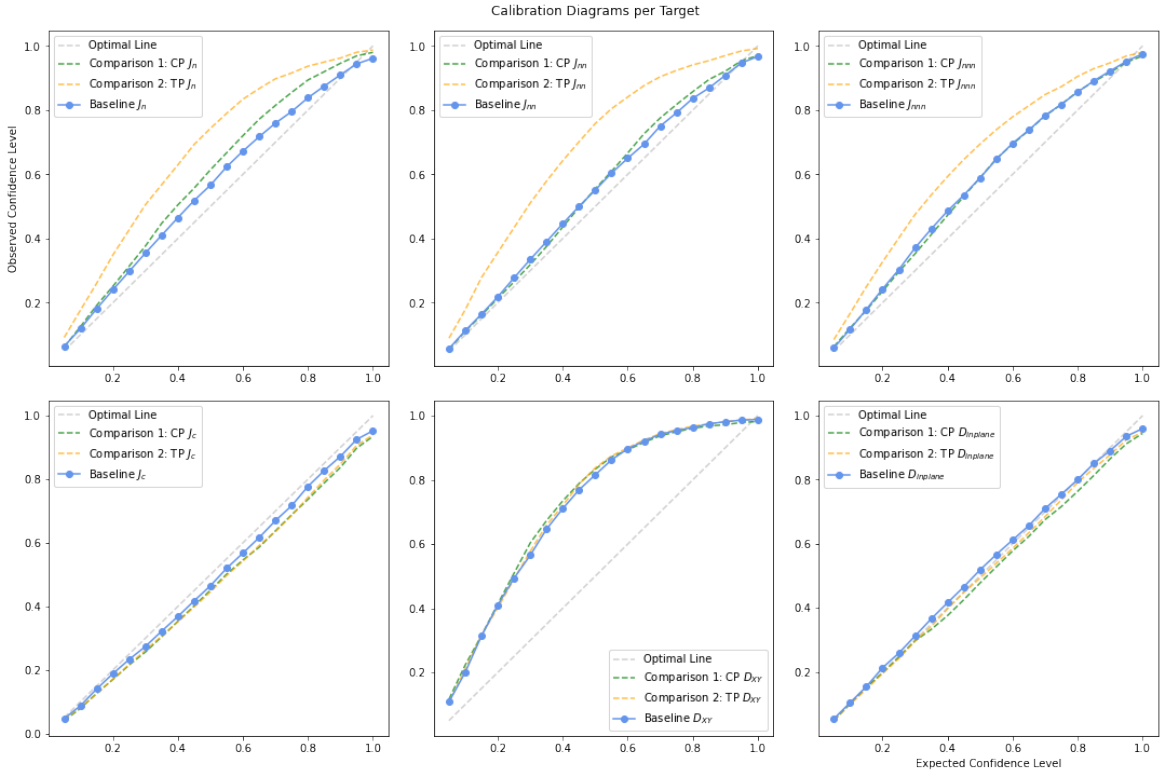

Figure 20: Comparison of the per target calibration curves of the Baseline model (BL) vs. Tempered Posterior model (TP) vs. Cold Posterior model (CP)

This is especially true for the tempered posterior model, which increased the over-all calibration loss by a factor of ~ 2.4, while the cold posterior model only increased the cal error by $\sim 26\%$. Table 2 also shows that the sources of deterioration of the calibration are the same parameters for which the accuracy could be improved for the tempered posterior, indicating the existence of an accuracy-vs-calibration trade-off when leveraging the CPE through simply down-weighting the complexity cost. Also, while the CP model's calibration curves are much closer to the BL model's curves, both modifications show a tendency to exaggerate the BL model's direction of miscalibration as can be seen in figure 20.

Lastly tempered and cold posteriors outperform the baseline on all targets regarding the sharpness of the predictions, as shown in the last three rows of table 2, with the CP model being slightly sharper than the TP model.

Considering all of the findings above, the cold posterior model can be considered a significant improvement over the baseline model, because it strongly improves upon accuracy and sharpness while almost preserving the original model's calibration.

Metric	Per Target Values						Over-all
	J_n	J_{nn}	J_{nnn}	J_c	D_{xy}	$D_{inplane}$	
MSE BL	0.00328	0.00475	0.00346	0.01182	0.00035	0.00711	0.00513
MSE TP	0.00054	0.00073	0.00094	0.01170	0.00018	0.00682	0.00348
MSE CP	0.00081	0.00121	0.00111	0.01185	0.00020	0.00681	0.00366
Cal BL	0.00249	0.00130	0.00419	0.00076	0.05040	0.00021	0.05934
Cal TP	0.03065	0.03360	0.01914	0.00285	0.05553	0.00043	0.14220
Cal CP	0.00793	0.00244	0.00407	0.00318	0.05622	0.00103	0.07486
Sha BL	0.00458	0.00688	0.00547	0.01277	0.00063	0.00873	0.03906
Sha TP	0.00131	0.00175	0.00167	0.01161	0.00024	0.00761	0.02418
Sha CP	0.00125	0.00170	0.00155	0.01112	0.00020	0.00697	0.02278

Table 2: Performance Comparison of the Baseline model (BL) vs. Tempered Posterior model (TP) vs. Cold Posterior model (CP). All values are rounded to five decimals.

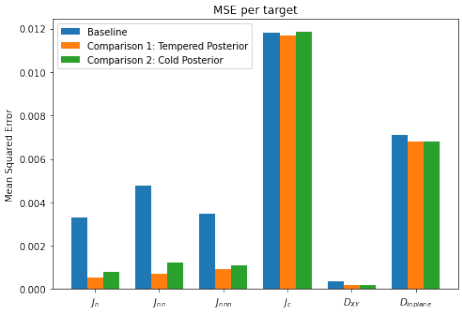

Figure 21: Comparison of the per target MSE scores of the Baseline model (BL) vs. Tempered Posterior model (TP) vs. Cold Posterior model (CP).

5.4 Experiment 3: Hybrid-Frequentist-Bayesian CNN

In order to model each weight as being a sample from some distribution, rather than having a single unknown ground truth value, BNNs require a significant amount of additional trainable parameters, which depends on the chosen distributions' parameterizations.

Even in the simplest case, where each parameter is modelled by its own independent Gaussian distribution, like in many implementations of GMFVI[14], BNNs require twice as many weights, one for the mean and one for the standard deviation of the Gaussian, as a frequentist model with the same architecture.

In an attempt to reduce the number of trainable parameters, while maintaining the ability to express uncertainty in the model, the following experiment explores the effectiveness of an approach inspired from the experiments conducted by J. Snoek et al.[40] and J. Zeng et al.[48].

5.4.1 Goal of the Experiment

In this experiment it is empirically tested whether a hybrid BNN can provide comparable performance to a fully-Bayesian NN on the given data set.

5.4.2 Architecture / Functional Model

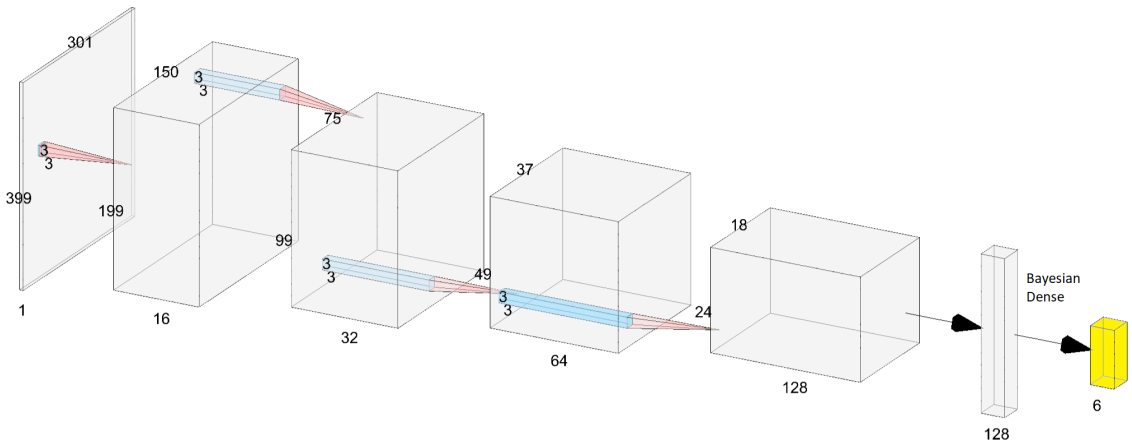

Figure 22: Architecture of the Hybrid BNN. All blocks are equal to those described in figure 13 except for the number of hidden units. Only the last layer is probabilistic.

Motivated by the approach presented by J. Zeng et al.[48], in a first attempt all but the last Bayesian dense layer of the baseline model have been replaced by point-estimate layers, while all other aspects have been kept equal to the BL's architecture. Training

this hybrid model with an adapted learning rate, found by the BHPO algorithm, however did not provide meaningful results, with the model converging to the mean-predictor during training.

The issue mentioned above proves that in general a fully-Bayesian NN can not be converted to a hybrid BNN by simply replacing Bayesian by frequentist layers. For this reason, in addition to the learning rate, for this experiment the number of blocks and the number of hidden units in the first layer, which determines the number of hidden units in all but the last other layers, have been adapted using BHPO, resulting in the architecture shown in figure 22 with a total of $\sim$ 100.000 trainable parameters.

5.4.3 Training

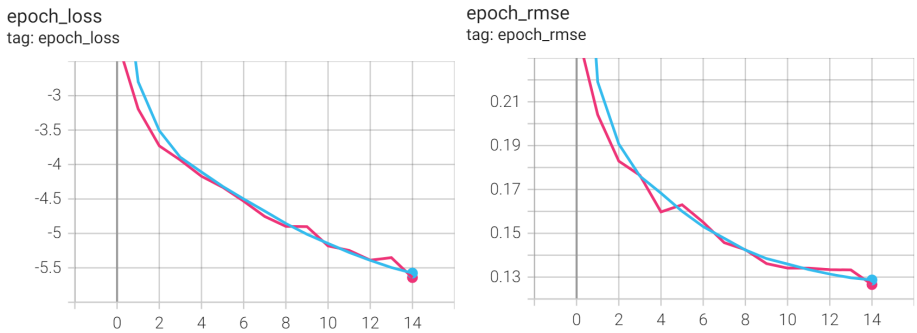

Figure 23: Overview: Progress of the training procedure. Left: VFE loss per epoch. Right: RMSE per epoch. Blue curve represents the corresponding quantity on the training split, magenta on the validation split.

Model training was done until convergence of the RMSE (figure 23), using a learning rate of 0.00031444 for a total of 15 epochs, with all other aspects of the training procedure left equal to the baseline training.

To prevent the point-estimate layers from over-fitting the L2 regularization penalty is applied for each point-estimate kernel w_i.

$$\mathrm{L2}(w_i) = \alpha * \sum_{j=1}^{K} (w_i^j)^2, \quad with \quad \alpha = 0.01 \tag{5.16}$$

5.4.4 Evaluation

According to the over-all scores in table 3 the BL model significantly outperforms the hybrid model on the MSE and Sha metrics, meaning the baseline predictions are sharper and more accurate than the predictions made by the hybrid model.

Metric	Per Target Values						Over-all
	J_n	J_{nn}	J_{nnn}	J_c	D_{xy}	$D_{inplane}$	
MSE BL	0.00328	0.00475	0.00346	0.01182	0.00035	0.00711	0.00513
MSE H	0.00881	0.00721	0.00911	0.01218	0.00297	0.00844	0.00812
Cal BL	0.00249	0.00130	0.00419	0.00076	0.05040	0.00021	0.05934
Cal H	0.00111	0.00056	0.00085	0.00108	0.00548	0.00099	0.01008
Sha BL	0.00458	0.00688	0.00547	0.01277	0.00063	0.00873	0.03906
Sha H	0.00867	0.00753	0.00933	0.01156	0.00283	0.00914	0.04906

Table 3: Performance Summary of the BL model vs. Hybrid model (H). All values are rounded to five decimals.

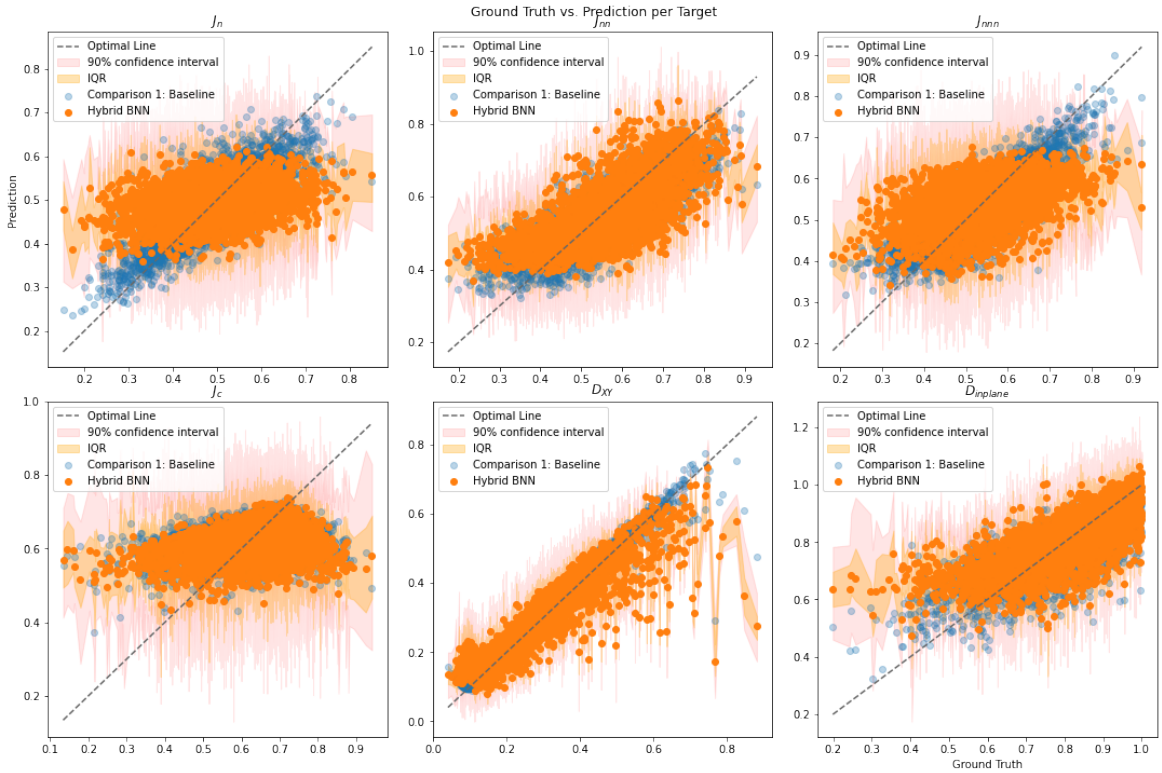

Figure 24: Ground Truth vs. BL Predictions vs. Hybrid Model Predictions.

Visually comparing the predictions against each other, as done in figure 24, confirms the interpretation drawn from the metrics. Except for J_c for which both models' point-clouds are almost equal, the point-clouds of the hybrid model's predictions are slightly broader and their orientations, especially for the parameters J_n and J_{nnn}, differ more from the optimal 45-degree line.

When it comes to calibration, the over-all Cal scores suggest that the hybrid BNN is better calibrated than the baseline BNN, with the nearest-neighbor interaction param-

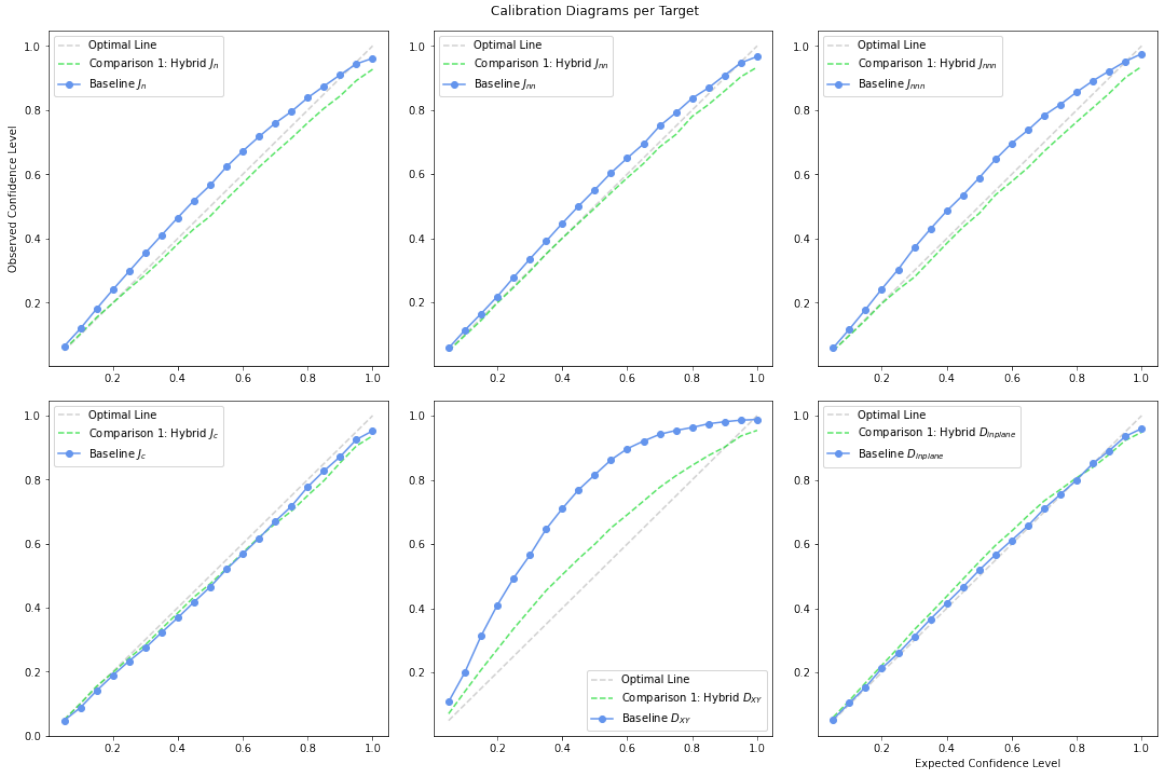

Figure 25: Calibration Curves: BL Model vs. Hybrid Model.

eters and easy-plane anisotropy D_{xy} being the main contributors to the performance gain. In contrast to the BL model, which is under-confident for most targets, the hybrid model has a weak tendency towards being over-confident for the majority of targets, which can be derived from figure 25.

In summary the results of the experiment, together with the overview of the training procedures tried during BHPO given in figure 26, imply that on the down-side hybrid BNNs are very sensitive to the choice of hyperparameters and therefore require additional effort to find an architecture matching the accuracy and sharpness of fully-Bayesian NNs.

On the up-side they provide a significant reduction of trainable parameters, e.g. in the case of this experiment by a factor of ~ 17, and more reliable uncertainty estimations when compared to fully-Bayesian models.

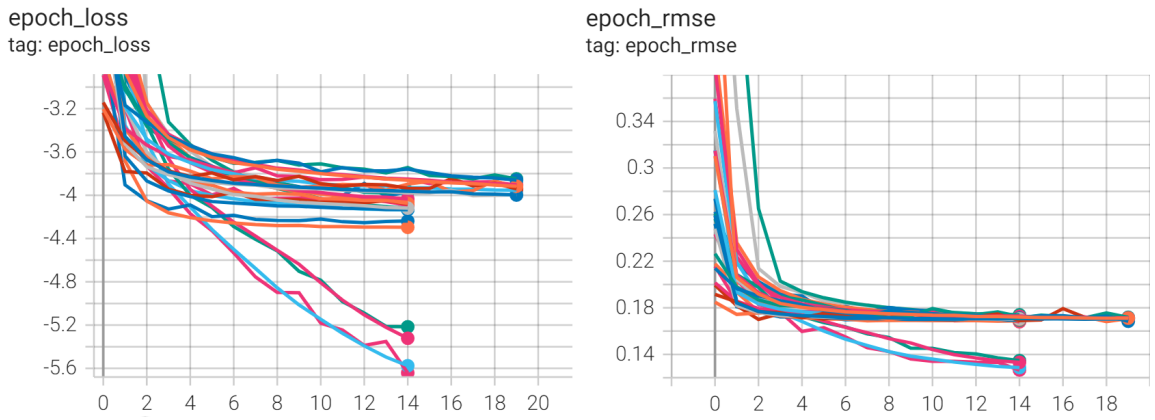

Figure 26: Overview: Training procedures run by the BHPO algorithm. Each colored line represents one hyperparameter configuration. Diverging configurations where excluded to improve visibility. Configurations trained for more than 15 epochs represent runs executed before the number of epochs was reduced to improve upon computation time while still reaching convergence.

5.5 Experiment 4: Combining All Features

5.5.1 Goal of the Experiment

This experiment was designed to empirically show how using the entire set of features during training affects the performance of the resulting model compared to the baseline.

5.5.2 Training

All other things equal the baseline model was trained from scratch using all seven cuts, padded to the same dimensions and concatenated channel-wise, as inputs.

5.5.3 Evaluation

Metric	Per Target Values						Over-all
	J_n	J_{nn}	J_{nnn}	J_c	D_{xy}	$D_{inplane}$	
MSE BL	0.00328	0.00475	0.00346	0.01182	0.00035	0.00711	0.00513
MSE AC	0.00128	0.00117	0.00068	0.00017	0.00011	0.00022	0.00060
Cal BL	0.00249	0.00130	0.00419	0.00076	0.05040	0.00021	0.05934
Cal AC	0.02365	0.01995	0.06231	0.00376	0.07299	0.05153	0.23418
Cal AC-RC	0.00060	0.00056	0.00084	0.00044	0.00092	0.00068	0.00405
Sha BL	0.00458	0.00688	0.00547	0.01277	0.00063	0.00873	0.03906
Sha AC	0.00224	0.00203	0.00084	0.00031	0.00044	0.00070	0.00655

Table 4: Performance Summary of the BL model vs. All-Cuts model (AC). All values are rounded to five decimals. AC-RC denotes the calibration scores of the AC model after applying the recalibration algorithm suggested in subsection 5.5.4 - Recalibration

Summarizing the results presented in table 4, the AC model strongly improves upon the BL model's predictions w.r.t. the accuracy and sharpness on all targets in general and especially on the inter-honeycomb-plane interaction parameter J_c.

Plotting the two models' predictions together against the ground truth values (figure 27) further reveals that, in contrast to the BL-BNN, the AC-BNN learned a meaningful model of the influence of J_c by leveraging the additional information encoded in the new features, explaining the strong accuracy and sharpness gains on this target.

Another finding from table 4 is that compared to the baseline, the AC model is poorly calibrated for all targets.

5.5.4 Recalibration

Motivated by the otherwise strong performance of the AC model, in this section a simple algorithm to retrospectively improve the calibration of a BNN, derived from

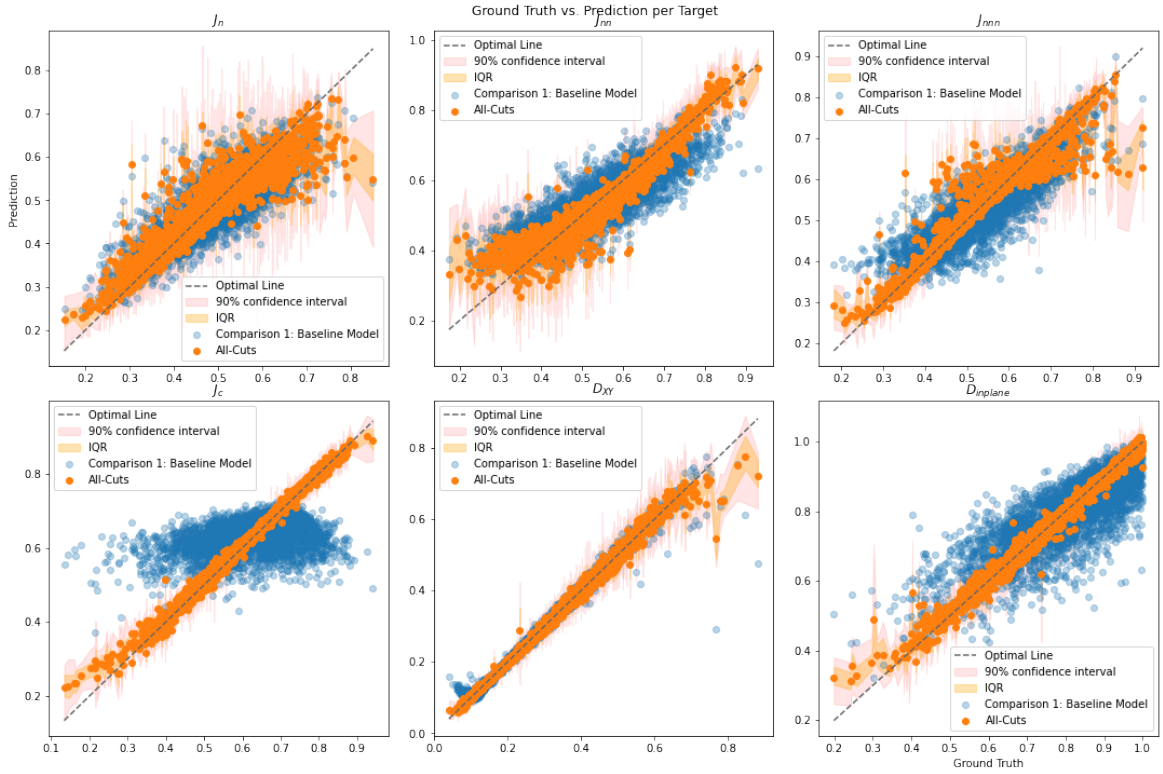

Figure 27: Ground Truth vs. Mean Predictions of the Baseline model (blue) and the All-Cuts model (orange) sorted in ascending order of ground truth y_i.

the method described by V.Kuleshov et al.[25], is suggested to alleviate the observed deterioration of the model's calibration.

Improving the calibration can be formally framed as a minimization problem of the over-all calibration error defined in section 5.2.5, which is a hierarchical sum consisting of the individual errors of all points on the per target calibration curves as visualized in figure 29. Therefore a mapping $f^{t*}(P_i)$, assigning each point $P_i^* = (p_i, p_i)$ on the optimal curve to a point on the actual calibration curve $\hat{P}_i^{\,t} = (p_i, \hat{p}_i^{\,t})$ with minimal squared distance between their empiric CIs —

$$\text{err}(P_i^*, \hat{P}_i^{\,t}) = w_i * (p_i - \hat{p}_i^{\,t})^2$$

, creates a recalibrated curve that minimizes the per-target calibration loss. Applying such a mapping to every targets' calibration curve then minimizes the over-all calibration error.

Following this reasoning the optimization problem can be re-framed to finding such an optimal set of mappings, where $\mathcal{F}^{all}$ denotes the set of all possible mappings.

$$\mathcal{F}^* = \{f^{1*}, ..., f^{T*}\} = \arg \min_{\{f^1,...,f^T\} \in \mathcal{F}^{all}} \sum_{t=1}^{T} \sum_{i=1}^{m} \text{err}(P_i^*, f^t(\hat{P}_i^t)) \tag{5.17}$$

The suggested algorithm to find $\mathcal{F}^*$ is comprised of the following steps:

1. Calculate the set of calibration curves

   $$\{\mathcal{C}^t = [(p_1, \hat{p_1}^t), ..., (p_m, \hat{p_m}^t)] | t \in \{1, ..., T\}\}$$

   for the uncalibrated model on a separate split of the data set, called calibration split
2. Define a distance metric that satisfies equation 5.17, e.g.

   $$\text{dist}(P1, P2) = (P1_y - P2_y)^2$$

3. For each target variable train a nearest-neighbor classifier, using the points in their respective calibration curve as training data, with the previously defined metric

Applying this recalibration scheme to the AC model leads to the calibration scores given in the AC-RC row of table 4, which underline the effectiveness of the suggested approach. After recalibration the resulting AC-RC model outperforms or closely matches the baseline's calibration performance on all targets.

Figure 28 shows that, while being less smooth, the improved calibration curves are closer to the optimal line and exhibit no clear tendency towards either of its sides.

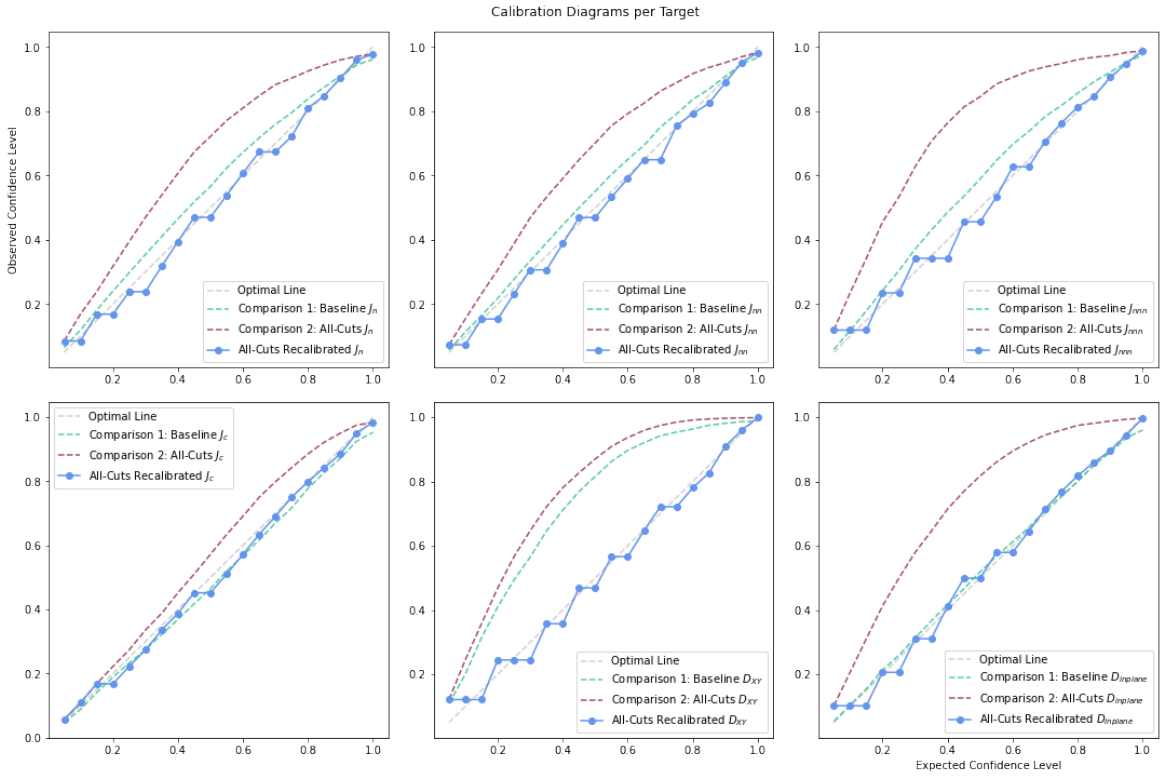

Figure 28: Comparison: Per-target calibration of BL model vs. AC model vs. AC-RC model.

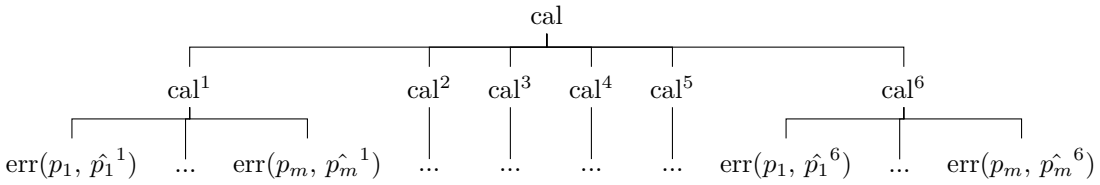

Figure 29: Composition-Tree of the over-all calibration error. Leaf nodes represent the contribution of an individual point on the respective calibration curve to the per target calibration loss as defined in section 5.2.5. Each parent node is a sum of all its child nodes.

6 Discussion

The purpose of this section is to discuss which conclusions can be drawn by considering all evaluation results of the previous experiments with a focus on how these results relate to the research questions stated in section Previous Work and Addressed Challenges and which limitations need to be considered.

The question of whether ANNs, employed in an end-to-end manner, are capable of providing meaningful estimates of Hamiltonian parameters from the given simulated data set is best discussed by considering the results provided by experiments Experiment 1: Baseline Bayesian CNN, Experiment 2: Influence of the Cold Posterior Effect and Experiment 4: Combining All Features.

Taken in isolation, the results of experiment 4 show that sharp and accurate predictions for the targets J_c, D_{XY} and $D_{inplane}$ can be obtained, with a comparably high level of confidence, by providing the network with all defined cuts simultaneously. Getting estimates of the same quality for the nearest-neighbor interaction parameters on the other hand appears to be less straight forward with J_n being the parameter with the widest spread around the ground truth.

Although other explanations can not be entirely ruled out, a reasonable hypothesis for the performance gap between the two sets of parameters is that the cuts provided as input to the network do not contain the information required for the network to learn to tell apart the influence of the nearest-neighbor parameters. Besides the general argument that the cuts only represent a small fraction of the reciprocal space and are therefore likely to miss important information, support for this hypothesis is given by two observations.

The first observation is that the predictions for J_{nn} and J_{nnn} are just as sharp and accurate as those for the first set of parameters for most values and only exhibit intervals for which the performance deteriorates. These intervals are coherent, rather than random, and not located at the edges of the possible value ranges and are therefore not likely to be caused by artefacts of the parameter sampling procedure. This leads to the conclusion that there exist certain regimes of values for which isolating the influences of the parameters is harder than for other regimes.

Second, when compared to the results obtained in experiment 1, it can be observed that adding more cuts improved upon the performance on all targets with the effect on the nearest-neighbor parameters just being less pronounced. If adding more information from the reciprocal space allows for very tight predictions on the other parameters the same should be true for the nearest-neighbor interaction parameters.

For the reasons above it is suggested to provide additional cuts in the simulated data set, specifically chosen to provide complementary information about J_n, J_{nn} and J_{nnn} in increasing order of significance.

Another important take-away in this context is that it appears to be advisable to either

construct the training data set by sampling from a uniform, as for example done by A. M. Samarakoon et al.[37], rather than a Gaussian distribution. Support for this claim is given by the simple observation that across all experiments the performance of the models significantly drops in areas where very few training samples exist.

Regarding the question of trade-offs to consider during modelling, no evidence was found during experimentation that very deep network architectures are required to achieve the best performance. Indeed, although models with more hidden layers have been tried by the BHPO algorithm, the best-performing models in experiments 1 and 3 ended up being models with four to five hidden layers. Given that these networks are rather shallow and in accordance with the previous findings that incorporating as much information from the reciprocal space as possible seems to be key to accurately regress the targets, it is therefore suggested to keep the full resolution of the inputs if enough computational resources are available, instead of further increasing the depth of the network.

To further reduce the computational requirements it has also been demonstrated in Experiment 3: Hybrid-Frequentist-Bayesian CNN that the number of trainable parameters can be significantly reduced by employing hybrid instead of fully-bayesian neural networks and that these networks are capable of extracting meaningful patterns from the data to a certain extend. However the model's performance in experiment 3 is not satisfactory when compared to the other models suggested in this work and should therefore not be used on an *as-is* basis. A possible approach to tackle this problem might be to train a fully-frequentist model in a first step and then, after replacing the deterministic head by a probabilistic one and freezing all other weights, retrain the model until convergence to disentangle the tasks of learning high level features and uncertainty estimates in analogy to the *adaptive basis regression* approach used by J. Snoek et al.[40].

Finally another trade-off has been demonstrated in experiment 2, where it was shown how to leverage the CPE to strongly improve upon a BNN's sharpness and accuracy at the cost of only slightly deteriorating the calibration. Employing this method also comes at the cost of reduced mathematical interpretability because it requires to artificially sharpen the posterior.

Concerning the question of how uncertainty estimation, quantitative or qualitative, is performed in hybrid approaches, the first common method is to use some kind of quality threshold to obtain a set of likely candidate predictions for a given task. For example in a closely related paper A. M. Samarakoon et al. use an empirically found threshold on their defined similarity measures between the real and the simulated data or their respective latent space projections.[37] Although easy to interpret the resulting estimates are heavily influenced by the experience of the researchers and a suitable method has to be found on a case-to-case basis depending on the used algorithm.

The alternative, commonly deployed method is to independently train a set of frequentist models and use the set of their individual predictions on the same input to obtain

uncertainty estimates. Downsides of this approach are that one has to decide on the total number of models and their respective architectures in the set, again on a case-to-case basis, and train all models until convergence, leading to potentially large training times.

BNNs, which can be understood as an infinite set of models with the same architecture trained within a finite amount of time, are a special case of the latter approach, which allows incorporating a stochastic model into a NN and by doing so strikes a balance between training time and expressiveness of the resulting uncertainty estimates.

As such, BNNs also come with some of the same disadvantages, for example the need to specify a computationally feasible stochastic model over the hard to interpret weights of an ANN under severe computational constraints. As the results in the conducted experiments confirm, this is not a trivial task because a misspecified stochastic model prevents the NN from learning well calibrated uncertainty estimates. For example in the case of the AC model, for the sake of isolating the effect of providing additional information per sample on the model's performance, all other aspects of the model have been left equal in section 5.5. It is however reasonable to assume that the additional information given by the new features and their correlations should lead to a generally lower level of epistemic uncertainty. Consequentially it is likely that the prior picked for this experiment is not well chosen and that a prior with lower variance might improve upon the calibration of the AC model's strongly under-confident predictions.

To provide a more generally applicable approach, in section 5.5.4 - Recalibration an algorithm has been suggested to recalibrate a BNN model. On the upside it has been demonstrated that, using this algorithm, close-to-optimal calibration can be achieved, meaning that the resulting uncertainty estimates of the model are very reliable, without the need to retrain the BNN or adapt its stochastic model. Although effective and simple to implement, a downside of this approach is that, in contrast to an arguably harder to obtain well-specified stochastic model, it does not a have a meaningful probabilistic interpretation.

Answering the last open question, regarding which measures effectively reduce uncertainty of predictions, in sections 5.2.6 - Influence of Training-Split Size on Performance and 5.5 - Experiment 4: Combining All Features it has been empirically confirmed that, in accordance with theory, the model's epistemic uncertainty can be reduced, with diminishing returns, by increasing the number of training samples and by including more information about the phenomenon to be modelled.

7 Conclusion and Future Work

The first contribution of this work to the current efforts of using INS data to estimate the parameters of a given Hamiltonian lies in the deployment of Bayesian methods to incorporate the concept of epistemic uncertainty into NN-based approaches. Having reliable estimations of uncertainty in the model's parameters provides researchers with valuable information to avoid common pitfalls occurring when working with standard neural networks.

For example in the case of a frequentist NN, that was successfully trained on simulated data and then presented with a sample of a real measurement, a frequentist network will confidently output a single prediction of the target parameters. Given that real measurements, due to noise and artefacts, can look very different from simulated ones and might therefore lie outside of the distribution the model was trained on, the trustworthiness of these predictions is questionable. In contrast to this behaviour, when presented with an out-of-distribution sample or a sample that does not contain the information required to infer the target variable, a Bayesian neural network model will output predictions with a certain degree of variance, indicating how confident the model is about its predictions as it was shown in the *Evaluation* section of Experiment 1: Baseline Bayesian CNN.

Furthermore through the experiments conducted in sections Experiment 1: Baseline Bayesian CNN, Experiment 2: Influence of the Cold Posterior Effect and Experiment 4: Combining All Features it has been shown that a shallow convolutional neural network, optimized using Bayesian hyperparameter optimization, is capable of accurately regressing the target variables from a collection of cuts through the reciprocal space at full resolution and that the cold posterior effect can be leveraged to further improve the performance of a BNN model at the cost of reducing mathematical interpretability.

Finally in section Experiment 3: Hybrid-Frequentist-Bayesian CNN a method has been suggested to reduce the number of trainable parameters required for modelling epistemic uncertainty at the cost of accuracy and a more extensive hyperparameter search.

Regarding areas for further improvement of the model, the question of how to deal with noise and artefacts found in real experimental data is of special importance, since, as already mentioned above, NN models in general do not perform well on samples that differ too much from their training data. Therefore to be able to gain meaningful predictions from real experimental data two broad categories of approaches can be identified:

1. **Adapting the Simulation**: All methods that intend to close the gap between real and simulated data by incorporating more experimental details into the training data. Examples of this method include the usage of masks to cover areas of the reciprocal space that are not observed during the experiment and the incorporation of instrument specifications into the simulation such as spatial

and energetic resolution.

2. **Adapting the Experimental Data**: All methods that intend to close the gap between real and simulated data by removing noise and artefacts from a specific sample of real experimental data. Among others this method includes using expert knowledge to manually identify and remove distortions.

Whether a single approach or a combination of both should be used can not be answered in general and depends on the specifics of the use case.

Another area of interest is concerned with the question of how to obtain calibrated models without resorting to retrospective recalibration algorithms. This aspect is of importance because models that learn reliable uncertainties from the data possess a stochastic model that is in better agreement with the true distribution of the data and for this reason can be considered as being more trustworthy.

Lastly investigating how to close the performance gap between fully and hybrid BNNs has the potential to further drive progress in the field, because the comparatively low number of trainable parameters per layer would make deeper and more complicated architectures more feasible.

8 Bibliography

References

[1] Laurence Aitchison. *A statistical theory of cold posteriors in deep neural networks*. 2020. DOI: 10.48550/ARXIV.2008.05912. URL: https://arxiv.org/abs/2008.05912.

[2] Christopher M Bishop. *Pattern Recognition and Machine Learning*. Ed. by M Jordan, J Kleinberg, and B Schölkopf. Vol. 4. Information science and statistics 4. Springer, 2006. Chap. Graphical, p. 738. ISBN: 9780387310732. DOI: 10.1117/1.2819119. arXiv: 0-387-31073-8. URL: http://www.library.wisc.edu/selectedtocs/bg0137.pdf.

[3] David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. "Variational Inference: A Review for Statisticians". In: *Journal of the American Statistical Association* 112.518 (Apr. 2017), pp. 859–877. DOI: 10.1080/01621459.2017.1285773. URL: https://doi.org/10.1080%2F01621459.2017.1285773.

[4] Charles Blundell et al. *Weight Uncertainty in Neural Networks*. 2015. DOI: 10.48550/ARXIV.1505.05424. URL: https://arxiv.org/abs/1505.05424.

[5] D Changwoo, W-R Chen, and S Lee. "Small angle scattering data analysis assisted by machine learning methods". In: *MRS Adv.* 5 (2020), pp. 1577–84. DOI: 10.1557/adv.2020.130. URL: https://doi.org/10.1557/adv.2020.130.

[6] François Chollet et al. *Keras*. https://keras.io. 2015.

[7] Beau Coker, Weiwei Pan, and Finale Doshi-Velez. *Wide Mean-Field Variational Bayesian Neural Networks Ignore the Data*. 2021. DOI: 10.48550/ARXIV.2106.07052. URL: https://arxiv.org/abs/2106.07052.

[8] *Crystallography edited by Walter Borchardt-Ott*. eng. 2nd ed. 1995. Berlin, Heidelberg, 1995. ISBN: 9783642577543.

[9] Joshua V. Dillon et al. *TensorFlow Distributions*. 2017. DOI: 10.48550/ARXIV.1711.10604. URL: https://arxiv.org/abs/1711.10604.

[10] Mathieu Doucet et al. "Machine learning for neutron scattering at ORNL *". In: *Machine Learning: Science and Technology* 2 (2 June 2021), p. 023001. DOI: 10.1088/2632-2153/abcf88.

[11] Stefan Elfwing, Eiji Uchibe, and Kenji Doya. "Sigmoid-weighted linear units for neural network function approximation in reinforcement learning". In: *Neural Networks* 107 (2018). Special issue on deep reinforcement learning, pp. 3–11. ISSN: 0893-6080. DOI: https://doi.org/10.1016/j.neunet.2017.12.012. URL: https://www.sciencedirect.com/science/article/pii/S0893608017302976.

[12] Sebastian Farquhar, Lewis Smith, and Yarin Gal. *Liberty or Depth: Deep Bayesian Neural Nets Do Not Need Complex Weight Posterior Approximations*. 2020. DOI: 10.48550/ARXIV.2002.03704. URL: https://arxiv.org/abs/2002.03704.

[13] Randy S. Fishman. *Spin-wave theory and its applications to neutron scattering and THz spectroscopy / Randy S. Fishman, Jaime A. Fernandez-Baca, Toomas Rõõm*. eng. IOP (Series). Release 5. San Rafael [California] 40 Oak Drive, San Rafael, CA, 94903, USA, 2018. ISBN: 9781643271149.

[14] Vincent Fortuin et al. *Bayesian Neural Network Priors Revisited*. 2021. DOI: 10.48550/ARXIV.2102.06571. URL: https://arxiv.org/abs/2102.06571.

[15] C Garcia-Cardona et al. "Learning to predict material structure from neutron scattering data". In: *2019 IEEE Int. Conf. Big Data (Big Data)*. 2019, pp 4490–7.

[16] Ethan Goan and Clinton Fookes. "Bayesian Neural Networks: An Introduction and Survey". In: *Case Studies in Applied Bayesian Data Science*. Springer International Publishing, 2020, pp. 45–87. DOI: 10.1007/978-3-030-42553-1_3. URL: https://doi.org/10.1007%2F978-3-030-42553-1_3.

[17] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[18] David J. Griffiths and Darrell F. Schroeter. *Introduction to Quantum Mechanics*. 3rd ed. Cambridge University Press, 2018. DOI: 10.1017/9781316995433.

[19] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. "Multilayer feedforward networks are universal approximators". In: *Neural Networks* 2.5 (1989), pp. 359–366. ISSN: 0893-6080. DOI: https://doi.org/10.1016/0893-6080(89)90020-8. URL: https://www.sciencedirect.com/science/article/pii/0893608089900208.

[20] Christian Janiesch, Patrick Zschech, and Kai Heinrich. "Machine Learning and Deep Learning". In: *Electronic Markets* 31.3 (2021), pp. 685–695. DOI: 10.1007/s12525-021-00475-2.

[21] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2014. DOI: 10.48550/ARXIV.1412.6980. URL: https://arxiv.org/abs/1412.6980.

[22] Diederik P. Kingma, Tim Salimans, and Max Welling. *Variational Dropout and the Local Reparameterization Trick*. 2015. DOI: 10.48550/ARXIV.1506.02557. URL: https://arxiv.org/abs/1506.02557.

[23] Diederik P. Kingma and Max Welling. "Auto-Encoding Variational Bayes". In: *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*. 2014. arXiv: http://arxiv.org/abs/1312.6114v10 [stat.ML].

[24] E. S. Klyushina et al. “Investigation of the spin-1 honeycomb antiferromagnet BaNi2V2O8 with easy-plane anisotropy”. In: *Physical Review B* 96.21 (Dec. 2017). ISSN: 2469-9969. DOI: 10.1103/physrevb.96.214428. URL: http://dx.doi.org/10.1103/PhysRevB.96.214428.

[25] Volodymyr Kuleshov, Nathan Fenner, and Stefano Ermon. *Accurate Uncertainties for Deep Learning Using Calibrated Regression*. 2018. DOI: 10.48550/ARXIV.1807.00263. URL: https://arxiv.org/abs/1807.00263.

[26] Yann Lecun, Patrick Haffner, and Y. Bengio. “Object Recognition with Gradient-Based Learning”. In: (Aug. 2000).

[27] Chia-Hao Liu et al. “Using a machine learning approach to determine the space group of a structure from the atomic pair distribution function”. In: *Acta Crystallographica Section A* 75.4 (July 2019), pp. 633–643. DOI: 10.1107/S2053273319005606. URL: https://doi.org/10.1107/S2053273319005606.

[28] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[29] Martín Martínez-Ripoll. *Crystallography-Cristalografia*. 2014. URL: https://www.xtal.iqfr.csic.es/Cristalografia/index-en.html (visited on 03/08/2022).

[30] Kevin P. Murphy. *Machine learning : a probabilistic perspective*. Cambridge, Mass. [u.a.]: MIT Press, 2012. ISBN: 978-0-262-01802-9.

[31] *Neutron and X-ray Spectroscopy edited by Françoise Hippert, Erik Geissler, Jean Louis Hodeau, Eddy Lelièvre-Berna, Jean-René Regnard*. eng. Dordrecht, 2006. ISBN: 9781402033377.

[32] Lorenzo Noci et al. “Disentangling the Roles of Curation, Data-Augmentation and the Prior in the Cold Posterior Effect”. In: (2021). DOI: 10.48550/ARXIV.2106.06596. URL: https://arxiv.org/abs/2106.06596.

[33] Tom O’Malley et al. *KerasTuner*. https://github.com/keras-team/keras-tuner. 2019.

[34] Prajit Ramachandran, Barret Zoph, and Quoc V. Le. *Searching for Activation Functions*. 2017. DOI: 10.48550/ARXIV.1710.05941. URL: https://arxiv.org/abs/1710.05941.

[35] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005. ISBN: 026218253X.

[36] O Ronneberger, P Fischer, and T Brox. “U-net: convolutional networks for biomedical image segmentation”. In: *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015*. Ed. by N Navab et al. Springer Int. Publishing, 2015, pp. 234–41.

[37] Anjana M. Samarakoon et al. "Machine-learning-assisted insight into spin ice Dy2Ti2O7". In: *Nature Communications* 11.1 (Feb. 2020). ISSN: 2041-1723. DOI: 10.1038/s41467-020-14660-y. URL: http://dx.doi.org/10.1038/s41467-020-14660-y.

[38] David Silver et al. "Mastering the Game of Go with Deep Neural Networks and Tree Search". In: *Nature* 529.7587 (Jan. 2016), pp. 484–489. DOI: 10.1038/nature16961.

[39] Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. *Practical Bayesian Optimization of Machine Learning Algorithms*. 2012. DOI: 10.48550/ARXIV.1206.2944. URL: https://arxiv.org/abs/1206.2944.

[40] Jasper Snoek et al. *Scalable Bayesian Optimization Using Deep Neural Networks*. 2015. DOI: 10.48550/ARXIV.1502.05700. URL: https://arxiv.org/abs/1502.05700.

[41] Artur Souza et al. *DeepFreak: Learning Crystallography Diffraction Patterns with Automated Machine Learning*. 2019. DOI: 10.48550/ARXIV.1904.11834. URL: https://arxiv.org/abs/1904.11834.

[42] Nitish Srivastava et al. "Dropout: a simple way to prevent neural networks from overfitting." In: *Journal of Machine Learning Research* 15.1 (2014), pp. 1929–1958. URL: http://www.cs.toronto.edu/~rsalakhu/papers/srivastava14a.pdf.

[43] B Sullivan et al. "BraggNet: integrating Bragg peaks using neural networks". In: *J. Appl. Crystallogr.* 52 (2019), pp. 854–63. DOI: 10.1107/S1600576719008665. URL: https://doi.org/10.1107/S1600576719008665.

[44] S Toth and B Lake. "Linear spin wave theory for single-Q incommensurate magnetic structures". In: *Journal of Physics: Condensed Matter* 27.16 (Mar. 2015), p. 166002. DOI: 10.1088/0953-8984/27/16/166002. URL: https://doi.org/10.1088/0953-8984/27/16/166002.

[45] Yeming Wen et al. *Flipout: Efficient Pseudo-Independent Weight Perturbations on Mini-Batches*. 2018. DOI: 10.48550/ARXIV.1803.04386. URL: https://arxiv.org/abs/1803.04386.

[46] Florian Wenzel et al. *How Good is the Bayes Posterior in Deep Neural Networks Really?* 2020. DOI: 10.48550/ARXIV.2002.02405. URL: https://arxiv.org/abs/2002.02405.

[47] Robert M White. *Quantum Theory of Magnetism Magnetic Properties of Materials / by Robert M. White*. eng. Springer Series in SOLID-STATE SCIENCES, 32. Berlin, Heidelberg, 2007. ISBN: 9783540690252.

[48] Jiaming Zeng, Adam Lesnikowski, and Jose M. Alvarez. *The Relevance of Bayesian Layer Positioning to Model Uncertainty in Deep Bayesian Active Learning*. 2018. arXiv: 1811.12535 [cs.LG].

[49] Daniel Zhang et al. *The AI Index 2021 Annual Report*. 2021. DOI: 10.48550/ARXIV.2103.06312. URL: https://arxiv.org/abs/2103.06312.

9 Appendix

9.1 Table of Abbreviations

Abbreviation	Meaning
AC	All-Cuts
ANN	Artificial Neural Network
BHPO	Bayesian Hyperparameter Optimization
BL	Baseline
BNN	Bayesian Neural Network
CI	Confidence Interval
CNN	Convolutional Neural Network
CP	Cold Posterior
CPE	Cold Posterior Effect
DSF	Dynamical Structure Factor
ELBO	Evidence Lower Bound
FC	Fully Connected
GMFVI	Gaussian Mean Field Variational Inference
GP	Gaussian Process
INS	Inelastic Neutron Scattering
IQR	Interquartile Range
KL	Kullback-Leibler
LCB	Lower Confidence Bound
MFVI	Mean Field Variational Inference
ML	Machine Learning
MLE	Maximum-Likelihood Estimation
MLP	Multilayer Perceptron
MSE	Mean Squared Error
NN	Neural Network
RC	Recalibrated
RMSE	Root Mean Squared Error
ReLU	Rectified Linear Unit
SAS	Small-Angle-Scattering
SGD	Stochastic Gradient Descent
SGVB	Stochastic Gradient Variational Bayes
SLD	Scattering Length Density
SiLU	Sigmoid-Weighted Linear Unit
TP	Tempered Posterior
VAE	Variational Auto Encoder
VFE	Variational Free Energy
VI	Variational Inference

Selbständigkeitserklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig verfasst und noch nicht für andere Prüfungen eingereicht habe. Sämtliche Quellen einschließlich Internetquellen, die unverändert oder abgewandelt wiedergegeben werden, insbesondere Quellen für Texte, Grafiken, Tabellen und Bilder, sind als solche kenntlich gemacht. Mir ist bekannt, dass bei Verstößen gegen diese Grundsätze ein Verfahren wegen Täuschungsversuchs bzw. Täuschung eingeleitet wird.

Berlin, den 08.10.2022