The Care Label Concept:
A Certification Suite for Trustworthy and
Resource-Aware Machine Learning

Katharina Morik[1], Helena Kotthaus[1], Lukas Heppe[1], Danny Heinrich[1],
Raphael Fischer[1], Andreas Pauly[1], and Nico Piatkowski[2]

[1] TU Dortmund University, Germany
{firstname}.{lastname}@tu-dortmund.de
[2] Fraunhofer IAIS, Germany
nico.piatkowski@iais.fraunhofer.de

Abstract. Machine learning applications have become ubiquitous. This has led to an increased effort of making machine learning trustworthy. Explainable and fair AI have already matured. They address knowledgeable users and application engineers. For those who do not want to invest time into understanding the method or the learned model, we offer *care labels*: easy to understand at a glance, allowing for method or model comparisons, and, at the same time, scientifically well-based. On one hand, this transforms descriptions as given by, e.g., Fact Sheets or Model Cards, into a form that is well-suited for end-users. On the other hand, care labels are the result of a certification suite that tests whether stated guarantees hold.

In this paper, we present two experiments with our certification suite. One shows the care labels for configurations of Markov random fields (MRFs). Based on the underlying theory of MRFs, each choice leads to its specific rating of static properties like, e.g., expressivity and reliability. In addition, the implementation is tested and resource consumption is measured yielding dynamic properties.

This two-level procedure is followed by another experiment certifying deep neural network (DNN) models. There, we draw the static properties from literature on a particular model and data set. At the second level, experiments are generated that deliver measurements of robustness against certain attacks. We illustrate this by ResNet-18 and MobileNetV3 applied to ImageNet.

Keywords: Trustworthy AI · Green AI · Resource-Aware AI · Certification

1 Introduction

Since machine learning (ML) has become a prime enabling technology for diverse application areas, several academic fields and society discuss capabilities and limitations of artificial intelligence (AI). Aspects range from safety [20], accountability

[1], and responsibility [8] to general ethical questions [10]. Trustworthy AI answers the societal demands [2] and surveys methods for increasing trustworthiness [23] or proposes AI internal audits [41]. Since there are diverse related parties in each ML application, diverse methods are needed for each stakeholder type [29]. Explanations of methods or learned models allow application specialists a detailed interaction with system developers [12,44]. Recently, the need of documenting interactions with respect to fairness has been put forward [46]. Approaches to auditing and explanations are important for interested parties in the application area and society. Interactive modeling of applications has a long tradition within ML [34]. All these attempts demand a considerable amount of preoccupation with ML methods and models, however not everybody has the time or is inclined to dig that deep. Hence, in this paper, we describe a different undertaking.

For users who want to deploy a learned model in a similar way as they use their kitchenware or electrical equipment, we adapt *care labels* [35,5], that turn knowledge about ML systems into guarantees for their use. We enhance the approach via certification tests. Our framework is schematically displayed in Figure 1. On the one hand, care labels may be considered smart transformations of ML process descriptions or documentations, that are customary. ML pipelines together with all hyperparameter settings are visually presented at different levels of abstractions by tools like, e.g., RapidMiner [32], where also short characteristics of used methods are available as text. For deep neural networks (DNNs), which are pipelines in their own right, Fact Sheets [42] and Model Cards [33] allow for documentation of learned models. Hyperparameters of training experiments and the model together with training and testing data are described in textual form. Several repositories store experimental results and corresponding code, data and learned model[3]. This is all made for developers and, indeed, we also use these sources of knowledge about models. In contrast, we construct ratings of properties according to literature and communicate them in form of A(best) to D(worst).

On the other hand, care labels express properties based on underlying theory of ML methods. Each class of methods introduces specific categories that assign proven properties to its algorithms for training and inference. There are several algorithms for learning decision trees, support vector machines, K-Means clustering, to name just a few. For many variants, theoretical results exist that prove potentials and limitations of the method. Representing this knowledge in the form of meta-data allows for an automated characterization and rating, which is presented by the care label. In this paper, the class of exponential families illustrates our approach. It shows how we exploit configuration options of a method. A Markov random field (MRF) is configured by choice of a loss function, inference algorithm, and optimizer. Each configuration yields a specific care label based on underlying theory, we show them for MRFs in section 4.

Finally, the static characteristics of a method or model are not enough. It is necessary to test, whether the properties hold for a given implementation. Here, we enter the field of certification [7], especially that of verifiable claims [3].
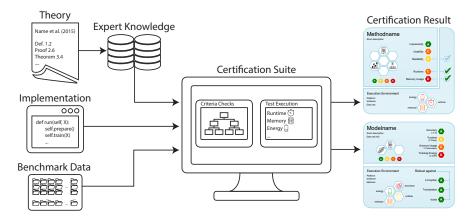
---

[3] https://paperswithcode.com

Fig. 1: Certification suite for trustworthy and resource-aware machine learning

Moreover, we want to also take into account the hardware platform. Particular properties refer to algorithms tailored to, e.g., CPU, GPU, microcontrollers, ultra-low power devices [40] or low voltage accelerators [48]. In addition to runtime and memory usage, energy consumption is measured and shown in the care label. For DNNs, robustness tests are categories of the dynamic care label part.

The contributions of this paper are:

- Novel presentation format for ML descriptions
    - easy-to-understand care label design,
    - criteria together with their rating in the respective value range.
- Theory-based characteristics of ML methods
    - exploit theoretical guarantees of ML algorithms, if tight bounds exist,
    - exploit published results of models on data sets, otherwise.
- Testing properties of ML
    - the characteristics are written in the form of meta-data and
    - meta-data drive the certification procedures.

In this paper, we report on related work in section 2, before we describe our approach in section 3. Extensive experiments with both MRFs and DNNs (section 4) show the broad range already covered. Section 5 concludes.

2   Related Work

The call for explainability has come along with the success of DNNs, cf. e.g., [44]. Many approaches exist for interaction of model developers and model deployers. Care labels summarizing properties at a glance, however, need some verification of properties. In the following, we first report on work on which we base care labels for DNNs. Second, we report on work allowing care labels for MRFs, since we should not forget about probabilistic or information-theoretic methods.

2.1   Verification and Testing of DNN

DNNs as universal function approximators are composed of layers, activation functions, and are ultimately defined by a large number of hyperparameters. Resulting prediction quality or robustness, in the contrary, is not compositional. Theoretical bounds on the sample complexity or learnability in the PAC framework often refer to the VC-dimension. Recently, tighter bounds of the VC-dimension of DNN have been proven for those using only the ReLU activation function [15]. Similarly, for ReLU networks, a verification procedure has been proposed using satisfiability modulo theories (SMT) [25]. They do not take into account, whether it is a convolutional DNN, a variational auto-encoder, or even a generative adversarial network (GAN).

While theoretical constructs such as neural tangent kernels [24] or eigenvalue computations [53] help in understanding the theoretical properties, they cannot yet be used to assign care labels to the variants of DNNs. The survey on verification and testing of DNNs [23] presents SMT solvers and tests that are based on efficiently determining interesting regions [22]. Reachability of true target values can be guaranteed through nested optimization[43]. General bounds do not differ for different types of DNNs, i.e. we cannot derive distinguishing property values from these theoretical results for classes of DNNs. More importantly the complexity of calculating properties based on this theory is prohibitive.

Hence, we do not assign care labels to particular types of the DNN method. Instead, we have to base our care labels for DNNs on trained *models* of particular architectures and data sets. This is very close to Fact Sheets [42] and Model Cards [33], but different in the presentation and verification of properties. Testing methods for trained models based on neuron coverage [14] or benchmarking [18] are useful. Research has focused on verifying robustness and generalization capabilities of DNNs using carefully crafted corruptions and attacks specifically tailored towards a single use case. Szegedy et al. [49] have shown that considering robustness against small perturbations, so-called adversarial attacks, is crucial in the verification process. A game-theoretic approach to achieve guarantees for DNNs is based on a maximum safe radius regarding the attack [52]. Types of adversarial attacks are introduced by [27,51,6], and provided by tools like CleverHans [37].

The energy consumption especially for training a DNN is an important criterion and needs to be shown on a care label. The estimated CO_2 consumption has been listed for NLP training [47]. On the one hand, energy-efficient architectures are needed for model deployment on, e.g., mobile phones and trained models are specialized for different hardware architectures [4]. On the other hand, methods for estimating the energy consumption for deep learning are proposed [11,17]. For the care labels, we make good use of available tools.

2.2   Probabilistic Graphical Models

In this paper, we work with MRFs which are theoretically well-founded probabilistic graphical models. Their generative nature allows utilization for complex

tasks [9], and they have been adapted for distributed learning [19] and for systems under tight resource constraints [39].

Given a vector of random variables $\boldsymbol{X} = (X_1, X_2, \ldots, X_n)$, MRFs model the joint probability density and can be expressed according to the Hammersley-Clifford theorem [13] by a graph $G = (V, E)$ factorizing over the set of maximal cliques:

$$\mathbb{P}(X = x) = \frac{1}{Z} \prod_{C \in \mathcal{C}(G)} \psi_C(x_C) \ , \tag{1}$$

where $\psi_c(x_c)$ is a non-negative potential function parametrized by some weights $\boldsymbol{\theta} \in \mathbb{R}^n$ and Z is the partition function ensuring proper normalization

$$Z = \sum_{x \in \mathcal{X}} \prod_{C \in \mathcal{C}(G)} \exp\left(\langle \boldsymbol{\theta}_C, \phi(x)_C \rangle\right). \tag{2}$$

Here we set $\psi_c(x_c) = \exp\left(\langle \boldsymbol{\theta}_c, \phi(x)_c \rangle\right)$ to the canonical exponential family with clique-wise one-hot encoded sufficient statistics $\phi(x)$. Since $\mathbb{P}$ factorizes over the cliques, $\boldsymbol{\theta}$ is the concatenation of individual clique parameters.

Given a data set $\mathcal{D}$, weights $\boldsymbol{\theta}$ can be estimated by specifying a loss function and an optimizer, e.g., first-order gradient-based methods on log-likelihood. Therefore, an optimizer and a loss function have to be specified. Common choices are gradient descent and likelihood function. Given a set of parameters $\boldsymbol{\theta}$ we can perform probabilistic inference, e.g., perform queries for marginal probabilities or maximum-a-posteriori states. Probabilistic inference is #P-complete in general. A prominent representative of exact methods is the junction tree (JT) algorithm [30], which yields exact results on arbitrary graphs at cost of exponential runtime and memory cost in size of the largest clique. To overcome this limitation, several algorithms for approximate inference such as loopy belief propagation (LBP) were proposed [36], which trade guarantees against saving resources. As we shall see, MRF care labels benefit from theoretical results.

Early model-checking approaches for verifying the accuracy of probabilistic models have been presented [28]. These refer to other types of models, namely Markov chains, Markov decision processes, and probabilistic automata. Further work could use the insights there for another set of care labels.

3 Care Label Certification Suite

For certifying the behavior of ML systems and inform end-users about their properties, we propose a certification suite, visualized in Figure 1. It draws from three inputs: the expert knowledge database, a given ML implementation, and benchmark data. The expert knowledge consists of *static* criteria based on scientific work and theoretical results, both are used to generate tests and ratings (criteria checks). Furthermore, the implementation and benchmark data are used to run performance tests and assign ratings to *dynamic* properties, e.g., energy consumption per prediction (test performance). Following this two-level procedure, the suite provides the user with care labels, which summarize the findings and

test results for certification and comparison. We start by discussing our design concept for care labels. We then explore how their content is generated and tested, namely the static (subsection 3.2) and dynamic properties (subsection 3.3).

3.1 Design Concept

We propose two care label designs as displayed in Figure 1, one for particular trained models (*models* for short), and one for the methods with tight bounds that can be efficiently computed (*methods* for short). Our design allows non-experts to understand and compare different ML systems regarding their trustworthiness and resource consumption. Care labels are subdivided into two segments, informing about static (*Method-, Modelname*) and dynamic (*Execution Environment*) properties. Inspired by well-known certificates such as energy or Nutri-score labels, we denote to what extend certain properties are exhibited via an easy-to-understand four level rating system A (best) – D (worst). Similar to textile care labels, our labels display intuitive badges for noteworthy properties, e.g., a badge shaped like a *mobile phone* indicating applicability for resource-constrained systems.

The upper parts include meta information and ratings that can either be statically computed or extracted from literature. Information in the lower parts (e.g., robustness or memory consumption) stem from executing the implementation of the method or model on a specific hardware platform. Thus, the content of the lower parts might change dependent on the execution environment. The certification suite performs static checks based on expert criteria and dynamic tests based on the implementation and its execution environment. In the following sections, respective properties, tests, and ranking procedures are introduced.

3.2 Static Properties: Expert Criteria

For processing expert knowledge, we developed a meta-data driven certification procedure, visualized in Figure 2. For each *Component* of a method, e.g., JT as an inference algorithm choice for discrete MRFs (see Figure 4), we denote static properties via expert criteria that are grouped into the categories *Resources*, *Expressivity*, *Reliability*, and *Usability*. The last three categories are rated based on composed *Rules*, e.g., for simple cases as combination of fulfilled / not fulfilled criteria. Certain fulfilled criteria can also directly trigger *Checks* or *Tests* for certifying implementation properties.

The category *Resources* is either derived from theoretical asymptotic worst-case scenarios for methods or determined by a dynamic ranking. For ML *Methods* and components, the rating for runtime and memory usage, w.r.t. input data dimension is reported on a worst-case *Complexity* scale (big $\mathcal{O}$-notation). For ML *Models* (see Figure 3), the resource category is rated via a relative task-specific ranking, based on statically extracted resource requirements. As the model complexity of DNN architectures typically reduces to a constant, we instead report and rate the memory usage via number of *Parameters* and runtime via number of floating point operations (FLOPS). In addition, we provide the energy
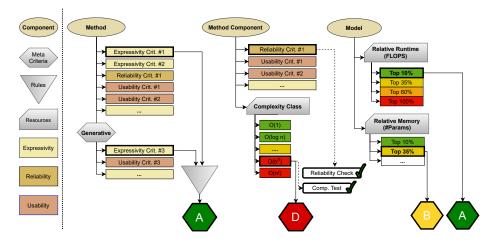
Fig. 2: Schematic representation of the meta-data driven certification procedure. For different ML components, sets of expert criteria (meta-data grouped into categories) are assessed. By applying expert rules and scales to the criteria the final rating is generated.

consumed by a model during training to provide an idea about expenses and environmental impact.

3.3   Dynamic Properties: Execution Environment

The dynamic properties of ML methods and models are either certified by checks that are generated based on expert criteria, or consist of execution environment specific performance measurements. This eases the comparison of different ML implementations regarding resource consumption. For the ML methods (see Figure 4), green check marks denote whether theoretical bounds on resource consumption and reliability hold. Unfortunately, checks on reliability as well as complexity class checks only partially apply to DNNs. However, the runtime and memory requirements for DNNs, especially for inference tasks, can be directly derived from the number of layers, parameters and floating point operations, independently from the execution environment. Instead, we verify DNNs using robustness tests previously mentioned in section 2. For the models (see Figure 3) the A-D ratings denote the relative performance of robustness tests.

We argue that for certification, most expert criteria that trigger implementation tests should be model or method-specific. Thus, we present the MRF and DNN-specific tests in section 4. In the following we discuss the details of rating and testing runtime and memory complexity for ML methods.

Runtime and memory complexity can commonly be derived from the underlying algorithmic requirements of a certain method. While the computation complexity is usually given in worst-case $\mathcal{O}$-notation, implementations may incur large coefficients on the asymptotic runtime. However, it is also possible that

implementations may not adhere to the given asymptotic runtime due to overhead or wrong implementation of an algorithm. To *verify runtime and memory costs* one can scale the given n to obtain a set of experiments that can be used to fit a set of curves e.g. linear $f(x) = ax + b$, quadratic $f(x) = ax^2 + bx + c$ and exponential functions $f(x) = a \cdot \exp(bx) + c$ to obtain the curve that contributes most to fitting the original memory or runtime data y for some scaling input x. Here we minimize the error for a set of selected complexity classes $\mathcal{F}$:

$$\arg\min_{f \in \mathcal{F}} \left\{ \min_{\boldsymbol{\theta}} \sum_{\boldsymbol{x} \in \mathcal{D}} ||y - f(\boldsymbol{x}, \boldsymbol{\theta})||_2^2 + \lambda R(\boldsymbol{\theta}) \right\} \tag{3}$$

This check is used in subsection 4.2 to verify the implementation's theoretical runtime and memory consumption.

4 Experiments: Applying the Care Label Certification

In this section, we apply our certification suite to DNNs and MRFs in order to illustrate our approach. We combine the robustness tests for DNNs mentioned in section 2 with our general concept presented in section 3 to showcase the care label concept.

4.1 Deep Neural Networks

In our experiments we focus on inference tasks, as the application of learned models is a common entry point for developing or extending new use cases. Properties of a neural network depend on the specific model architecture and thus also on the data set. Therefore, DNN care labels should always be issued for a specific combination of data set and model architecture.

For our experiments, we chose the PyTorch framework, since it provides pre-trained models via ModelZoo[4] together with the popular `ImageNet` data. To allow comparison and relative ranking, we ran experiments with four renowned models, namely AlexNet [26], `VGG11` [45], ResNet-18 [16], and MobileNetV3_Large [21]. For the two most recent models (ResNet-18 and MobileNetV3_Large), the final care labels are depicted in Figure 3. Note that more extensive experiments on a wider range of models would improve and refine our care label ranking results.

Static Properties In the upper segments of the care label (see Figure 3), statically extracted information like accuracy, number of gigaFLOPS (runtime), number of parameters (memory), and training energy consumption is presented for specific model architectures. To calculate the number of GFLOPS and parameters we used the `thop`[5] software package. Since the training energy cannot always be directly extracted from literature, e.g., via number of training epochs, we

---

[4] `https://pytorch.org/serve/model_zoo.html`
[5] `https://github.com/Lyken17/pytorch-OpCounter`

measured average wattage and epoch time for the respective models over ten epochs. These values are then multiplied with the number of epochs, required to reach target accuracy according to `TorchVision`[6].

Model (ImageNet)	Acc@Top1	Acc@Top5	Runtime (GFLOPS)	Memory (M Parameters)	Train Energy (kWh)
AlexNet [26]	56.52	79.07	1.43	61.10	**2.47**
VGG11 [45]	69.02	88.63	15.23	132.86	18.12
ResNet-18 [16]	69.76	89.08	3.64	11.69	6.55
MobileNetV3_Large [21]	**74.04**	**91.34**	**0.45**	**5.48**	35.13

Table 1: Static properties for different DNN architectures using ImageNet. For each column the best value is highlighted (rating A).

For the labels, we derive rankings (A-D) via relative comparison of static model information. We divide information for each property (i.e. columns of Table 1) into four quantiles and rate a given model based on the quantile it falls into. Thus, we award MobileNetV3_Large with an A for runtime, memory, and accuracy, scoring best in model comparison (ResNet-18 only receives B and C ratings). The labels however also indicate the downside of MobileNetV3_Large, as training energy is rated with a D. In fact, the required amount of energy is more than five times higher compared to the 6.55 kWh of ResNet-18 (awarded a B), due to the higher number of epochs (90 vs 600 epochs). The big advantage of MobileNetV3_Large is its suitability for mobile processors due to its modest parameter footprint, which the label illustrates via the *mobile phone* badge.

Dynamic Properties We investigate dynamic properties of the previously mentioned models during inference on different execution environments for CPU and GPU. All experiments have been executed on the same system, running Ubuntu 20.04 as operating system, equipped with an Intel Xeon W-2155 CPU, 64 GB of main memory, and an NVIDIA RTX 5000 GPU. For results on different execution platforms we use either the CPU alone or in combination with GPU acceleration. Table 2 and Figure 3 depict our findings, with runtime and energy consumption per predicted sample as well as maximum memory usage for a 3000 sample subset of ImageNet.

In addition, we performed a series of *robustness tests* using a set of task-specific attacks, as well as generic adversarial attacks. As task-specific attacks, we chose the ImageNet-C and ImageNet-P benchmarks, proposed by Hendrycks and Dietterich [18], where different *Perturbations* and *Corruptions* were applied at various severity levels. Additionally, we launched generic projected gradient descent (PGD) attacks, declared as *Noise* on our care labels. As proposed in [18] we evaluate our models using the relative mean corruption error (rel. mCE) on ImageNet-C and flip rate (FP) on ImageNet-P. Briefly, the rel. mCE measures the performance degradation when encountering different types of corruptions

---

[6] `https://github.com/pytorch/vision/tree/master/references/classification`

Model (ImageNet)	Robustness			Runtime (ms)		Memory (GB)		Energy (Ws)	
	Corr	Pert	Noise	CPU	CPU+GPU	CPU	CPU+GPU	CPU	CPU+GPU
AlexNet [26]	1.00	0.11	$0.24e^{-6}$	**10.52**	**1.24**	**0.65**	3.20+1.57	**0.51**	**0.05+0.07**
VGG11 [45]	1.18	**0.06**	**15.62e^{-6}**	122.95	2.36	1.73	3.49+3.11	5.99	0.10+0.32
ResNet-18 [16]	1.02	0.08	$0.12e^{-6}$	31.41	1.62	0.65	**3.06+1.43**	1.50	0.07+0.14
MobileNetV3_Large [21]	**0.94**	0.07	$0.49e^{-6}$	28.01	1.53	0.83	3.07+1.51	1.20	0.06+0.11

Table 2: Dynamic properties for various neural networks using ImageNet. For each column the best value is highlighted (rating A).

c at various severity levels s_c. It is computed as the average performance of a model f compared to its clean error E_{clean}^f normalized by some baseline b to allow for a fair comparison of different models, formally:

$$\text{rel. mCE} = \frac{1}{k}\sum_{c=1}^{k}\frac{1}{s_c}\frac{\sum_{i=1}^{s_c} E_{i,c}^f - E_{clean}^f}{\sum_{i=1}^{s_c} E_{i,c}^b - E_{clean}^b}. \tag{4}$$

The ImageNet-P benchmark is evaluated on a series of frame sequences. Later frames of a sequence were subjected to stronger perturbations. This benchmark is evaluated via the flip rate (FP), which is defined as the empirical probability of a frame to be classified differently than its preceding frame, i.e., labels flipping between two frames:

$$FP = \frac{1}{m(n-1)}\sum_{i=1}^{m}\sum_{j=2}^{n}\mathbb{1}_{x_{j-1}^i \neq x_j^i} \tag{5}$$

To verify robustness against noise, we use the proposed multi-step procedure proposed by [31] and provided by `CleverHans` [37] to create adversarial examples.

$$x^{t+1} = \Pi_{x+\epsilon}(x^t + \epsilon \cdot sgn(\nabla_x L(\theta, x, y))) \tag{6}$$

We estimate $\arg\max_\varepsilon$ w.r.t to the L_∞ norm, such that $f(x+\epsilon) = f(x)$ for most $x \in D$. The estimation is performed similar to binary search, for a given ϵ we generate a set S of adversarial examples according to Equation 6 and evaluate the accuracy drop acc_{drop} from f on S. If the accuracy falls below a certain threshold, the value of ϵ is halved until convergence. In our experiments we used 10 steps, a threshold of 10^{-4}, and $\epsilon = 0.001$ as initial value for generating adversarial examples (Equation 6). We present all dynamically measured results in Table 2, and determine relative care labels ratings in the same way as for the static properties. The robustness results for ResNet-18 indicate that it should not be used in safety-critical areas (ratings C-D), because predictions are unstable against corruptions, perturbations, and noise. In contrast, MobileNetV3_Large is less vulnerable (ratings A-B), but also consumes more energy and memory. This is counterintuitive with respect to its static properties, which probably stems from using the pure PyTorch model, without any platform-related enhancements.
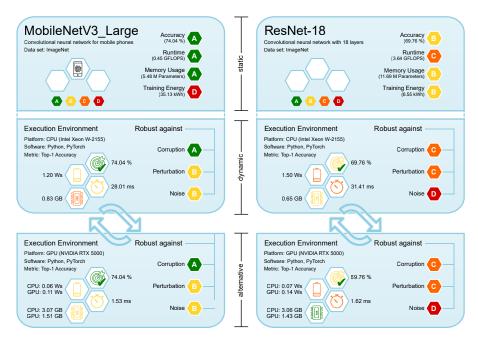
Fig. 3: Care labels for MobileNetV3_Large and ResNet-18 on different platforms.

We thus see that static properties like GFLOPS and number of parameters do not act as a clear indicator on how implementations behave dynamically [50].

Based on our ratings for robustness, the end-user gets important insights that DNNs suffer from high vulnerability against environmental influences and therefore shouldn't be applied in safety-critical areas. Thus, care labels allow end-users to quickly analyze possible opportunities and challenges when applying trained models.

4.2 Markov Random Fields

We provide care labels for MRFs in combination with either JT inference or LBP to highlight the implications caused by the algorithm choice. Our results in the form of care labels are shown in Figure 4.

Static Properties MRFs receive an *A* for *Expressivity* in both care labels (see Figure 4, upper part), since they can model any distribution of the exponential family. Besides, they provide uncertainty estimates and allow to solve a variety of probabilistic inference tasks. For *Usability*, the JT care label is awarded with *B*, while LBP is rated *C*. The reasoning here is that the underlying graph structure has to be either known or estimated for applying MRFs. The LBP label was assigned a lower rating due to requiring additional hyperparameters, namely convergence criteria for performing probabilistic inference. In terms of *Reliability*,

an A was determined for the JT configuration, since it is an deterministic algorithm and answers inference tasks exactly. As no such guarantees exist for LBP, a D was assigned. We here linked an underlying criteria to a reliability check, namely the *Probability Recovery Check*.

Looking at resource requirements, the JT care label was assigned a D rating for *Runtime* as well as *Memory Usage*, due to exponential scaling in the tree width [38]. In contrast, the LBP algorithm is quite resource friendly, since it grows at most quadratically in the number of states or edges. This behavior is also tested in the dynamic segment, which we now discuss in detail.

Dynamic Properties The dynamic results for our MRF variants are shown in the lower parts of Figure 4, each executed on two different environments (cf. subsection 4.1). As implementation we choose the pxpy library[7], since it provides implementations for different MRF configurations. For experiments, synthetic data was generated with grid graphs as underlying graph structure and binary states per vertex. We performed two types of checks, complexity class checks for memory and runtime bounds and reliability checks, both rewarded with check marks. In addition, we also report implementation depended resource requirements.

Reliability Checks: For certifying whether the implementation fulfills given reliability guarantees we used synthetic data, allowing for comparing the model output against known parameters. In the context of MRFs, we propose to perform a *Probability Recovery Check*. This test computes the true marginal distribution for given parameters by utilizing the exact JT algorithm, and then queries the provided implementation for its marginals under the given weights. Afterwards, nodewise Kullback-Leibler divergences between the two marginal vectors are computed and reduced to their max value. If this falls below an acceptance threshold ($10e - 3$), the check is considered passed.

Complexity Class Checks Here we test whether the execution behaviour on differently sized data sets scales with the theoretically given complexity classes (as explained in subsection 3.3). A grid graph with nodes varying from 2×2 up to 15×15 was used as input data. We show the underlying measurements for the runtime and memory bound checks on grid graphs in Figure 5. All complexity class checks have passed, since the behavior of the algorithms was in compliance with their theoretical memory and runtime bounds.

Similar to the DNN labels, we also report general resource measurements based on a specific data set (Grid 14×14). Here, the results highlight the differences in resource consumption with JT compared to LBP. For the JT algorithm we observe a trade-off between runtime and memory consumption when switching from CPU to GPU platform. Note that for LBP inference on GPUs, we observe a higher memory usage and runtime than for CPUs, due to I/O overhead caused by memory copy operations for small input sizes (for a larger input size this disadvantage would vanish).

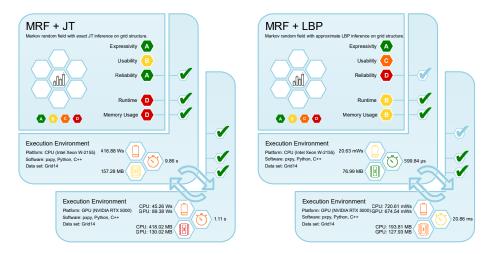[7] https://pypi.org/project/pxpy/

Fig. 4: Care labels for MRFs with different inference algorithms and platforms.

5   Conclusion

Today's ML technology affects diverse parties, such as application engineers and end-users. Having only little time to thoroughly study the theory, they instead require comprehensible certificates that build trust. Thus, a novel means of communicating information about ML is required, that has to address several points: Firstly, it has to be easily understandable for all stakeholders, and go beyond mere model documentation. Information on resource consumption is crucial for many application domains and needs to be transparently discussed. Aspects of certification should ideally be based on well-established theory, such as bounds and guarantees. Lastly, given ML implementations have to be tested, to assess their properties and check their compliance with underlying theory.

In this paper, we addressed all these aspects via care labels, obtained from our proposed certification suite. The labels were designed to be easily understandable, hiding the underlying intricacy, and allowing for quick comparison. We broke down the complexity of expert knowledge into meta-data criteria, enabling automated certification via label generation. We successfully applied our concept to two popular ML representatives, namely DNNs and MRFs. The labels for ResNet-18 and MobileNetV3_Large on ImageNet show how the choice of model architecture dramatically changes properties like accuracy, robustness, or resource consumption. For MRFs, our labels highlight the impact of choosing among probabilistic inference algorithms, which can either make the method more reliable, or more resource efficient.

This work proposed a novel format for ML certification to close the gap between experts and end-users. There is still room for further progress, by expanding on the expert knowledge, refining how meta information is handled and processed, and of course applying the concept to more methods and models. For
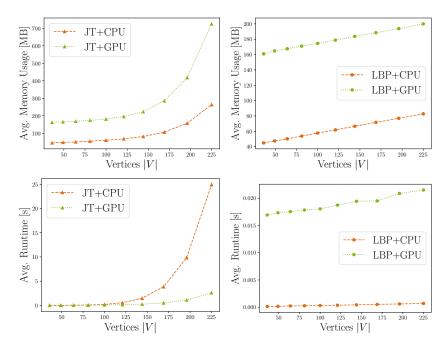
Fig. 5: We report memory [GB] and runtime [s] requirements for a MRF trained with `pxpy` on a grid graph with increasing grid size on GPU and CPU.

making certification via care labels universally accepted, we anticipate valuable feedback from our research colleagues, and hope that they join us in the endeavour of making ML more trustworthy and resource-aware.

References

1. Bellotti, V., Edwards, K.: Intelligibility and accountability: Human considerations in context-aware systems. Human-Computer Interaction **16**(2-4), 193–212 (2001)
2. Braunschweig, B., Ghallab, M.: Reflections on Artificial Intelligence for Humanity. Lecture Notes in Artificial Intelligence, Springer International Publishing (2021)
3. Brundage, M., et al.: Toward trustworthy ai development: Mechanisms for supporting verifiable claims (2020)

4. Cai, H., Gan, C., Wang, T., Zhekai, Z., Han, S.: Once for all: train one network and specialise it for efficient deployment. In: Procs. ICLR 2020 (2020)

5. Chatila, R., et al.: Trustworthy AI, chap. 2, pp. 18–45. Springer International Publishing (2021), `https://www.springer.com/de/book/9783030691271`

6. Chen, J., Jordan, M.I., Wainwright, M.J.: Hopskipjumpattack: A query-efficient decision-based attack. In: IEEE Symposium on Security and Privacy. IEEE (2020)

7. Cremers, A., et al.: Trustworthy use of artificial intelligence – priorities from a philosophical, ethical, legal, and technological viewpoint as a basis for certification of artificial intelligence. Fraunhofer Institute for Intelligent Analysis and Information Systems (IAIS) (2019), `https://www.iais.fraunhofer.de/content/dam/iais/KINRW/Whitepaper_Thrustworthy_AI.pdf`

8. Dignum, V.: Responsible Artificial Intelligence: How to Develop and Use AI in a Responsible Way. Springer (2019)

9. Fischer, R., Piatkowski, N., Pelletier, C., Webb, G., Petitjean, F., Morik, K.: No cloud on the horizon: Probabilistic gap filling in satellite image series. In: DSAA 2020. pp. 546–555

10. Floridi, L., et al.: AI4people?an ethical framework for a good ai society: Opportunities, risks, principles, and recommendations. Minds and Machines **28**(4) (2018)

11. García-Martín, E., Rodrigues, C.F., Riley, G., Grahn, H.: Estimation of energy consumption in machine learning. Journal of Parallel and Distributed Computing **134**, 75 − 88 (2019)

12. Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F., Pedreschi, D.: A survey of methods for explaining black box models. ACM Comput. Surv. **51**(5) (August 2018)

13. Hammersley, J.M., Clifford, P.: Markov fields on finite graphs and lattices (1971)

14. Harel-Canada, F., Wang, L., Ali Gulzar, M., Gu, Q., Kim, M.: Is neuron coverage a meaningful measure for testing deepneural networks? In: ESEC/FSE 2020: Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (November 2020)

15. Harvey, N., Liaw, C., Mehrabian, A.: Nearly-tight VC-dimension bounds for piecewise linear neural networks. In: Kale, S., Shamir, O. (eds.) Proceedings of the 2017 Conference on Learning Theory. Proceedings of Machine Learning Research, vol. 65, pp. 1064–1068. PMLR (2017)

16. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR 2016. pp. 770–778. IEEE Computer Society (2016)

17. Henderson, P., Hu, J., Romoff, J., Brunskill, E., Jurafsky, D., Pineau, J.: Towards the systematic reporting of the energy and carbon footprints of machine learning. Journal of Machine Learning Research **21**(248), 1–43 (2020)

18. Hendrycks, D., Dietterich, T.: Benchmarking neural network robustness to common corruptions and perturbations. ICLR (2019)

19. Heppe, L., Kamp, M., Adilova, L., Piatkowski, N., Heinrich, D., Morik, K.: Resource-constrained on-device learning by dynamic averaging. In: ECML PKDD 2020 Workshops. pp. 129–144. Springer International Publishing, Cham (2020)

20. Houben, S., et al.: Inspect, understand, overcome: A survey of practical methods for ai safety. Technical Report (2021), `https://www.ki-absicherung-projekt.de/fileadmin/KI_Absicherung/Downloads/KI-A_20201221_Houben_et_al_-_Inspect__Understand__Overcome.pdf`

21. Howard, A., et al.: Searching for mobilenetv3. In: ICCV 2019. pp. 1314–1324. IEEE (2019)

22. Huang, X., Kwiatkowska, M., Wang, S., Wu, M.: Safety verification of deep neural networks. In: International conference on computer aided verification. pp. 3–29. Springer (2017)
23. Huang, X., et al.: A survey of safety and trustworthiness of deep neural networks: Verification, testing, adversarial attack and defence, and interpretability. Computer Science Review **37**, 100270 (2020)
24. Jacot, A., Gabriel, F., Hongler, C.: Neural tangent kernel: Convergence and generalization in neural networks. In: Advances in Neural Information Processing Systems. vol. 31, pp. 8580–8589 (2018)
25. Katz, G., Barrett, C.W., Dill, D.L., Julian, K., Kochenderfer, M.J.: Reluplex: An efficient SMT solver for verifying deep neural networks. In: CAV 2017. vol. 10426, pp. 97–117. Springer (2017)
26. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems 25 (2012)
27. Kurakin, A., Goodfellow, Ian J.and Bengio, S.: Adversarial examples in the physical world. (ICLR) **5** (february 2017)
28. Kwiatkowska, M., Norman, G., Parker, D.: Prism 4.0: Verification of probabilistic real-time systems. In: Computer Aided Verification. pp. 585–591 (2011)
29. Langer, M., et al.: What do we want from explainable artificial intelligence (xai)? a stakeholder perspective on xai and a conceptual model guiding interdisciplinary xai research. Artificial Intelligence (Feb 2021)
30. Lauritzen, S.L., Spiegelhalter, D.J.: Local computations with probabilities on graphical structures and their application to expert systems. Journal of the Royal Statistical Society, Series B **50**(2), 157–224 (1988)
31. Madry, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A.: Towards deep learning models resistant to adversarial attacks (2019)
32. Mierswa, I., Wurst, M., Klinkenberg, R., Scholz, M., Euler, T.: YALE: Rapid Prototyping for Complex Data Mining Tasks. In: SIGKDD 2006. pp. 935–940
33. Mitchell, M., et al.: Model cards for model reporting. In: FAT* 2019. pp. 220–229. Association for Computing Machinery (2019)
34. Morik, K.: Balanced Cooperative Modeling, vol. 4, pp. 295–318. Morgan Kaufmann (1994)
35. Morik, K., et al.: Yes we care! – certification for machine learning methods through the care label framework (2021), arXiv preprint, arXiv:2105.10197
36. Murphy, K.P.: Machine Learning: a Probabilistic Perspective. MIT Press (2013)
37. Papernot, N., et al.: Technical report on the cleverhans v2.1.0 adversarial examples library (2018), `https://github.com/cleverhans-lab/cleverhans#readme`
38. Piatkowski, N.: Exponential Families on Resource-Constrained Systems. Ph.D. thesis, TU Dortmund University, Dortmund (2018), `https://eldorado.tu-dortmund.de/handle/2003/36877`
39. Piatkowski, N., Lee, S., Morik, K.: Integer undirected graphical models for resource-constrained systems. Neurocomputing **173**(1), 9–23 (January 2016)
40. Piatkowski, N., Sangkyun, L., Morik, K.: The integer approximation of undirected graphical models. In: ICPRAM 2014. pp. 296–304. SciTePress (2014)
41. Raji, I.D., et al.: Closing the ai accountability gap: Defining an end-to-end framework for internal algorithmic auditing. In: FAT* 2020. pp. 33–44
42. Richards, J., Piorkowski, D., Hind, M., Houde, S., Mojsilovic, A.: A methodology for creating AI factsheets (2020), arXiv preprint, arXiv:2006.13796
43. Ruan, W., Huang, X., Kwiatkowska, M.: Reachability analysis of deep neural networkswith provable guarantees. IJCAI 2018

44. Samek, W., Montavon, G., Vedaldi, A., Hansen, L.K., Müller, K.R. (eds.): Explainable AI: Interpreting, Explaining and Visualizing Deep Learning (2019)
45. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: ICLR 2015
46. Sokol, K., Flach, P.: Explainability fact sheets: a framework for systematic assessment of explainable approaches. In: FAT*. pp. 56–67. ACM (2020)
47. Strubell, E., Ganesh, A., McCallum, A.: Energy and policy considerations for modern deep learning research. In: AAAI 2020. pp. 13693–13696 (2020)
48. Stutz, D., Chandramoorthy, N., Hein, M., Schiele, B.: Bit error robustness for energy-efficient dnn accelerators. Proceedings of Machine Learning and Systems 3 pre-proceedings (MLSys 2021) (2020), accepted at MLSys 2021
49. Szegedy, C., et al.: Intriguing properties of neural networks. In: International Conference on Learning Representations (2014)
50. Tan, M., et al.: Mnasnet: Platform-aware neural architecture search for mobile. In: CVPR 2019. pp. 2820–2828 (2019)
51. Tramèr, F., Boneh, D.: Adversarial training and robustness for multiple perturbations. In: NeurIPS 2019. pp. 5858–5868 (2019)
52. Wu, M., Wicker, M., Ruan, W., Huang, X., Kwiatkowska, M.: A game-based approximate verification of deep neural networks with provable guarantees. Theor. Comput. Sci. **807**, 298–329 (2020)
53. Xiao, L., Pennington, J., Schoenholz, S.: Disentangling trainability and generalization in deep neural networks. In: ICML 2020. vol. 119, pp. 10462–10472