Make Sure You're Unsure: A Framework for Verifying Probabilistic Specifications

Leonard Berrada[*†] **Sumanth Dathathri** [*†] **Krishnamurthy (Dj) Dvijotham**[*†]

Robert Stanforth [†] **Rudy Bunel** [†] **Jonathan Uesato** [†] **Sven Gowal** [†] **M. Pawan Kumar** [†]

Abstract

Most real world applications require dealing with stochasticity like sensor noise or predictive uncertainty, where formal specifications of desired behavior are inherently probabilistic. Despite the promise of formal verification in ensuring the reliability of neural networks, progress in the direction of probabilistic specifications has been limited. In this direction, we first introduce a general formulation of probabilistic specifications for neural networks, which captures both probabilistic networks (e.g., Bayesian neural networks, MC-Dropout networks) and uncertain inputs (distributions over inputs arising from sensor noise or other perturbations). We then propose a general technique to verify such specifications by generalizing the notion of Lagrangian duality, replacing standard Lagrangian multipliers with "functional multipliers" that can be arbitrary functions of the activations at a given layer. We show that an optimal choice of functional multipliers leads to exact verification (i.e., sound and complete verification), and for specific forms of multipliers, we develop tractable practical verification algorithms.

We empirically validate our algorithms by applying them to Bayesian Neural Networks (BNNs) and MC Dropout Networks, and certifying properties such as adversarial robustness and robust detection of out-of-distribution (OOD) data. On these tasks we are able to provide significantly stronger guarantees when compared to prior work – for instance, for a VGG-64 MC-Dropout CNN trained on CIFAR-10 in a verification-agnostic manner, we improve the certified AUC (a verified lower bound on the true AUC) for robust OOD detection (on CIFAR-100) from $0\% \rightarrow 29\%$. Similarly, for a BNN trained on MNIST, we improve on the ℓ_∞ robust accuracy from $60.2\% \rightarrow 74.6\%$. Further, on a novel specification – distributionally robust OOD detection – we improve on the certified AUC from $5\% \rightarrow 23\%$.

1 Introduction

While neural networks (NNs) have shown significant promise in a wide-range of applications (for e.g., [He et al., 2016, Yu and Deng, 2014]), a key-bottleneck towards their wide-spread adoption in safety-critical applications is the lack of formal guarantees regarding safety and performance. In this direction, there has been considerable progress towards developing scalable methods that can provide formal guarantees regarding the conformance of NNs with desired properties [Katz et al., 2017, Dvijotham et al., 2018b, Raghunathan et al., 2018]. However, much of this progress has been in the setting where the specifications and neural networks do not exhibit any probabilistic behaviour, or is mostly specialized for specific probabilistic specifications [Weng et al., 2019, Wicker et al., 2020].

[*]Equal contribution. Authors listed in alphabetical order. Correspondance to lberrada@deepmind.com, sdathath@deepmind.com, dvij@cs.washington.edu.

[†]DeepMind, London, United Kingdom.

35th Conference on Neural Information Processing Systems (NeurIPS 2021), Sydney, Australia.

In contrast, we introduce a general framework for verifying specifications of neural networks that are probabilistic. The framework enables us to handle stochastic neural networks such as Bayesian Neural Networks or Monte-Carlo (MC) dropout networks, as well as probabilistic properties, such as distributionally robust out-of-distribution (OOD) detection. Furthermore, the specification can be defined on the output distribution from the network, which allows us to handle operations such as the expectation on functions of the neural network output.

Probabilistic specifications are relevant and natural to many practical problems. For instance, for robotics applications, there is uncertainty arising from noisy measurements from sensors, and uncertainty regarding the actions of uncontrolled agents (e.g. uncertainty regarding the behaviour of pedestrians for a self-driving vehicle). Often these uncertainties are modelled using a probabilistic approach, where a distribution is specified (or possibly learnt) over the feasible set of events [Thrun et al., 2005]. In such cases, we want to provide guarantees regarding the network's conformance to desired properties in the distributional setting (e.g. given a model of the pedestrian's uncertain behaviour, guarantee that the probability of collision for the autonomous vehicle is small). A more general problem includes scenarios where there is uncertainty regarding the parameters of the distribution used to model uncertainty. Here, in this general setting, we seek to verify the property that the network behaviour conforms with the desired specification under uncertainty corresponding to an entire set of distributions.

The key to handling the aforementioned complexity in the specifications being verified through our framework is the generalization of the Lagrangian duality. Specifically, instead of using the standard Lagrange duality where the multipliers are linear, we allow for probabilistic constraints (constraints between distributions) and use functional multipliers to replace the linear Lagrange multipliers. This allows us to exploit the structure of these probabilistic constraints, enabling us to provide stronger guarantees and facilitates the verification of *verification-agnostic networks* (networks that are not designed to be verifiable). In our paper, we focus on verification-agnostic networks as this is desirable for many reasons, as noted in Dathathri et al. [2020]. To summarize, our main contributions are:

- We derive a general framework that extends Lagrangian duality to handle a wide range of probabilistic specifications. Our main theoretical result (Theorem 1) shows that our approach (i) is always sound and computes an upper bound on the maximum violation of the specification being verified, and (ii) is expressive enough to theoretically capture tight verification (i.e. obtaining both sound and complete verification).
- We develop novel algorithms for handling specific multipliers and objectives within our framework (Propositions 1, 2). This allows us to apply our framework to novel specifications (such as distributionally robust OOD detection, where input perturbations are drawn from entire sets of distributions) by better capturing the probabilistic structure of the problem.
- We empirically validate our method by verifying neural networks, which are verification-agnostic, on a variety of probabilistic specifications. We demonstrate that even with relatively simple choices for the functional multiplier, our method strongly outperforms prior methods, which sometimes provide vacuous guarantees only. This further points towards the potential for significant improvements to be had by developing tractable optimization techniques for more complex and expressive multipliers within our framework.

2 Probabilistic Specifications

2.1 Notation

Let us consider a possibly stochastic neural network $\phi : \mathcal{X} \to \mathcal{P}(\mathcal{Y})$, where $\mathcal{X}$ is the set of possible input values to the model, $\mathcal{Y}$ is the set of possible output values, and $\mathcal{P}(\mathcal{Y})$ is the set of distributions over $\mathcal{Y}$. We assume that $\mathcal{Y}$ is a subset of $\mathbb{R}^l$ (unless specified otherwise), where l is the number of labels, and the output of the model are logits corresponding to unnormalized log-confidence scores assigned to the labels $\{1, \ldots, l\}$.

The model is assumed to be a sequence of K layers, each of them possibly stochastic. For $k \in \{1, \ldots, K\}$, $\pi_k(x_k|x_{k-1})$ denotes the probability that the output of layer k takes value x_k when its input value is x_{k-1}. We write $x_k \sim \pi_k(x_{k-1})$ to denote that x_k is drawn from the distribution over outputs of layer k given input x_{k-1} to layer k. We further assume that each $\pi_k(x)$ has the form $\sigma(\tilde{w}x + \tilde{b})$, where σ is a non-linear activation function (e.g., `ReLU`, sigmoid, `MaxOut`), and $\tilde{w}$ and $\tilde{b}$

are random variables. The stochasticity for layer π_k is assumed to be statistically independent of the stochasticity at other layers. For a BNN, $\tilde{w}$ and $\tilde{b}$ follow a diagonal Gaussian distribution (i.e., a Gaussian distribution with a diagonal covariance matrix), and for a MC-Dropout network they follow a Bernoulli-like distribution.

Given a distribution p_0 over the inputs $\mathcal{X}$, we use $\phi(p_0)$ to denote (with a slight abuse) the distribution of the random variable $\phi(X_0)$, where $X_0 \sim p_0$.

2.2 Problem Formulation.

We now introduce the general problem formulation for which we develop the verification framework.

Definition 1 (Probabilistic verification problem)**.** *Given a (possibly stochastic) neural network $\phi : \mathcal{X} \to \mathcal{P}(\mathcal{Y})$, a set of distributions over the input $\mathcal{P}_0$ and a functional $\boldsymbol{\psi} : \mathcal{P}(\mathcal{Y}) \mapsto \mathbb{R}$, the probabilistic verification problem is to check that the following is true:*

$$\forall\, p_0 \in \mathcal{P}_0,\ \boldsymbol{\psi}\left(\phi\left(p_0\right)\right) \leq 0. \tag{1}$$

2.3 Examples of Specifications

Below we provide examples of probabilistic specifications which are captured by the above problem formulation, and that we further empirically validate our framework on. In Appendix A, we provide further examples of relevant specifications (e.g., ensuring reliable uncertainty calibration) that can be handled by our problem setup.

Distributionally Robust OOD Detection. We consider the problem of verifying that a stochastic neural network assigns low confidence scores to all labels for OOD inputs, even in the presence of bounded noise perturbations to the inputs. Given a noise distribution perturbing an OOD image x_{ood}, we require that the expected softmax is smaller than a specified confidence threshold $p_{\max}$ for each label i. Since the precise noise distribution is most often unknown, we wish to consider an entire class $\mathcal{P}_{noise}$ of noise distributions. Denoting by δ_x the Dirac distribution around x, the problem is then to guarantee that for every p_0 in $\mathcal{P}_0 = \{\delta_{x_{ood}} + \omega : \omega \in \mathcal{P}_{noise}\}$ and for each possible label i, $\boldsymbol{\psi}(\phi(p_0)) \coloneqq \mathbb{E}_{y\sim\phi(p_0)}[\texttt{softmax}\,(y)_i] - p_{\max} \leq 0$. Robust OOD detection under bounded ℓ_∞ perturbations as considered in Bitterwolf et al. [2020] is a special case of this problem where $\mathcal{P}_{noise}$ is restricted to a set of δ distributions over points with bounded ℓ_∞ norm.

Robust Classification. We also extend the commonly studied robust classification problem [Madry et al., 2017] under norm-bounded perturbations, to the setting of probabilistic neural networks (e.g. BNNs). Define $\mathcal{P}_0$ to be the set of δ input distributions centered at points within an ϵ-ball of a nominal point x_{nom}, with label $i \in \{1, \ldots, l\}$: $\mathcal{P}_0 = \{\delta_x : \|x - x_{\text{nom}}\| \leq \epsilon\}$. For every $p_0 \in \mathcal{P}_0$, we wish to guarantee that the stochastic NN correctly classifies the input, i.e. for each j, $\boldsymbol{\psi}(\phi(p_0)) \coloneqq \mathbb{E}_{y\sim\phi(p_0)}[\texttt{softmax}\,(y)_i - \texttt{softmax}\,(y)_j] \leq 0$. Note that it is important to take the expectation of the softmax (and not logits) since this is how inference from BNNs is performed.

3 The Functional Lagrangian Framework

We consider the following optimization version:

$$\texttt{OPT} = \max_{p_0 \in \mathcal{P}_0} \boldsymbol{\psi}\left(\phi\left(p_0\right)\right), \tag{2}$$

Having $\texttt{OPT} \leq 0$ here is equivalent to satisfying specification (1) . However, solving problem (2) directly to global optimality is intractable in general, because it can possibly be a challenging nonlinear and stochastic optimization problem. However, to only verify that the specification is satisfied, it may suffice to compute an upper bound on OPT. Here, we describe how the functional Lagrangian framework allows to derive such bounds by decomposing the overall problem into smaller, easier sub-problems.

3.1 General Framework

Let $\mathcal{X}_k$ denote the feasible space of activations at layer k, and let p_k denote the distribution of activations at layer k when the inputs follow distribution p_0 (so that $p_K = \phi\left(p_0\right)$).

Assumptions. In order to derive our verification framework, we make the following assumptions:
(A1): $\exists\, l_0 \leq u_0 \in \mathbb{R}^n$ such that for each input distribution $p_0 \in \mathcal{P}_0$, Support $(p_0) \subseteq \mathcal{X}_0 = [l_0, u_0]$.
(A2): Each layer is such that if $x \in \mathcal{X}_k = [l_k, u_k]$, then Support $(\pi_k(x)) \subseteq \mathcal{X}_{k+1} = [l_{k+1}, u_{k+1}]$.

Assumption (A1) is natural since the inputs to neural networks are bounded. Assumption (A2) can be restrictive in some cases: it requires that the layer output is bounded with probability 1, which is not true, for example, if we have a BNN with a Gaussian posterior. However, we can relax this assumption to requiring that the output is bounded with high probability, as in Wicker et al. [2020].

Functional Lagrangian Dual. In order to derive the dual, we begin by noting that problem (2) can be equivalently written in the following constrained form:

$$\max_{p_0 \in \mathcal{P}_0, p_1, \ldots, p_K} \boldsymbol{\psi}(p_K) \text{ s.t. } \forall\, k \in \{0, \ldots, K-1\}, \forall\, y \in \mathcal{X}_{k+1},\ p_{k+1}(y) = \int_{\mathcal{X}_k} \pi_k(y|x)\, p_k(x)\, dx.$$

For the k-th constraint, let us assign a Lagrangian multiplier $\lambda_{k+1}(y)$ to each possible $y \in \mathcal{X}_{k+1}$. Note that $\lambda(y)$ is chosen independently for each y, hence λ is a *functional multiplier*. We then integrate over y, which yields the following Lagrangian penalty to be added to the dual objective:

$$-\int_{\mathcal{X}_{k+1}} \lambda_{k+1}(y)\, p_{k+1}(y)\, dy + \int_{\mathcal{X}_k, \mathcal{X}_{k+1}} \lambda_{k+1}(y)\, \pi_k(y|x)\, p_k(x)\, dx dy. \tag{3}$$

We now make two observations, which are described here at a high level only and are available in more details in appendix B. First, if we sum these penalties over k and group terms by p_k, it can be observed that the objective function decomposes additively over the p_k distributions. Second, for $k \in \{1, \ldots, K-1\}$, each p_k can be optimized independently (since the objective is separable), and since the objective is linear in p_k , the optimal p_k is a Dirac distribution, which means that the search over the probability distribution p_k can be simplified to a search over feasible values $x_k \in \mathcal{X}_k$. This yields the following dual:

$$\max_{p_K \in \mathcal{P}_K} \left(\boldsymbol{\psi}(p_K) - \int_{\mathcal{X}_K} \lambda_K(x)\, p_K(x)\, dx \right) + \sum_{k=1}^{K-1} \max_{x \in \mathcal{X}_k} \left(\int_{\mathcal{X}_{k+1}} \lambda_{k+1}(y)\, \pi_k(y|x)\, dy - \lambda_k(x) \right)$$
$$+ \max_{p_0 \in \mathcal{P}_0} \int_{\mathcal{X}_0} \left(\int_{\mathcal{X}_1} \lambda_1(y)\, \pi_0(y|x)\, dy \right) p_0(x)\, dx, \tag{4}$$

where we define $\mathcal{P}_K \triangleq \phi(\mathcal{P}_0)$. In the rest of this work, we refer to this dual as $g(\lambda)$, and we use the following notation to simplify equation (4):

$$g(\lambda) = \max_{p_0 \in \mathcal{P}_0} g_0(p_0, \lambda_1) + \sum_{k=1}^{K-1} \max_{x_k \in \mathcal{X}_k} g_k(x_k, \lambda_k, \lambda_{k+1}) + \max_{p_K \in \mathcal{P}_K} g_K(p_K, \lambda_K). \tag{5}$$

The dual $g(\lambda)$ can be seen as a generalization of Lagrangian relaxation [Bertsekas, 2015] with the two key modifications: (i) layer outputs are integrated over possible values, and (ii) Lagrangian penalties are expressed as arbitrary functions $\lambda_k(x)$ instead of being restricted to linear functions.

Main Result. Here, we relate the functional Lagrangian dual to the specification objective (2).

Theorem 1. *For any collection of functions $\lambda = (\lambda_1, \ldots, \lambda_K) \in \mathbb{R}^{\mathcal{X}_1} \times \ldots \times \mathbb{R}^{\mathcal{X}_K}$, we have that $g(\lambda) \geq$ `OPT`. In particular, if a choice of λ can be found such that $g(\lambda) \leq 0$, then specification (1) is true. Further, when $\boldsymbol{\psi}(p_K) = \mathbb{E}_{y \sim p_K}[c(y)]$, the dual becomes tight: $g(\lambda^\star) =$ `OPT` if $\lambda^\star$ is set to:*

$$\lambda_K^\star(x) = c(x)\,; \forall\, k \in \{K-1, \ldots, 1\},\ \lambda_k^\star(x) = \mathop{\mathbb{E}}_{y \sim \pi_k(x)} \left[\lambda_{k+1}^\star(y) \right].$$

Proof. We give a brief sketch of the proof - the details are in Appendix B. The problem in constrained form is an infinite dimensional optimization with decision variables $p_0, p_1, \ldots, p_K$ and linear constraints relating p_k and p_{k+1}. The Lagrangian dual of this optimization problem has objective $g(\lambda)$. By weak duality, we have $g(\lambda) \geq$ `OPT`. The second part of the theorem is easily observed by plugging in $\lambda^\star$ in $g(\lambda)$ and observing that the resulting optimization problem is equivalent to (2). □

Example. Let $\mathcal{P}_0$ be the set of probability distributions with mean 0, variance 1, and support $[-1, 1]$, and let $\mathcal{N}_{[a,b]}(\mu, \sigma^2)$ denote the normal distribution with mean μ and variance σ^2 with truncated support $[a, b]$. Now consider the following problem, for which we want to compute an upper bound:

$$\texttt{OPT} = \max_{p_0 \in \mathcal{P}_0} \mathbb{E}_{X_1}[\exp(-X_1)] \quad \text{s.t. } X_1|X_0 \sim \mathcal{N}_{[0,1]}(X_0^2, 1) \text{ and } X_0 \sim p_0. \tag{6}$$

This problem has two difficulties that prevent us from applying traditional optimization approaches like Lagrangian duality [Bertsekas, 2015], which has been used in neural network verification Dvijotham et al. [2018b]. The first difficulty is that the constraint linking X_1 to X_0 is stochastic, and standard approaches can not readily handle that. Second, the optimization variable p_0 can take any value in an entire set of probability distributions, while usual methods can only search over sets of real values. Thus standard methods fail to provide the tools to solve such a problem. Since the probability distributions have bounded support, a possible way around this problem is to ignore the stochasticity of the problem, and to optimize over the worst-case realization of the random variable X_1 in order to obtain a valid upper bound on OPT as: $\texttt{OPT} \leq \max_{x_1 \in [0,1]} \exp(-x_1) = 1$. However this is an over-pessimistic modeling of the problem and the resulting upper bound is loose. In contrast, Theorem 1 shows that for any function $\lambda : \mathbb{R} \to \mathbb{R}$, OPT can be upper bounded by:

$$\texttt{OPT} \leq \max_{x_1 \in [0,1], p_0 \in \mathcal{P}_0} \exp(-x_1) - \lambda(x_1) + \mathbb{E}_{X_0 \sim p_0}[\mathbb{E}_{X_1|X_0 \sim \mathcal{N}_{[0,1]}(X_0^2,1)}[\lambda(X_1)]].$$

This inequality holds true in particular for any function λ of the form $x \mapsto \theta x$ where $\theta \in \mathbb{R}$, and thus:

$$\begin{aligned}
\texttt{OPT} &\leq \inf_{\theta \in \mathbb{R}} \max_{x_1 \in [0,1], p_0 \in \mathcal{P}_0} \exp(-x_1) - \theta x_1 + \mathbb{E}_{X_0 \sim p_0}[\mathbb{E}_{X_1|X_0 \sim \mathcal{N}_{[0,1]}(X_0^2,1)}[\theta X_1]], \\
&= \inf_{\theta \in \mathbb{R}} \max_{x_1 \in [0,1], p_0 \in \mathcal{P}_0} \exp(-x_1) - \theta x_1 + \theta \mathbb{E}_{X_0 \sim p_0}[X_0^2], \\
&= \inf_{\theta \in \mathbb{R}} \max_{x_1 \in [0,1]} \exp(-x_1) - \theta x_1 + \theta \approx 0.37.
\end{aligned}$$

Here, our framework lets us tractably compute a bound on OPT that is significantly tighter compared to the naive support-based bound.

3.2 Optimization Algorithm

Parameterization. The choice of functional multipliers affects the difficulty of evaluating $g(\lambda)$. In fact, since neural network verification is NP-hard [Katz et al., 2017], we know that computing $g(\lambda^\star)$ is intractable in the general case. Therefore in practice, we instantiate the functional Lagrangian framework for specific parameterized classes of Lagrangian functions, which we denote as $\lambda(\theta) = \{\lambda_k(x) = \lambda_k(x; \theta_k)\}_{k=1}^K$. Choosing the right class of functions $\lambda(\theta)$ is a trade-off: for very simple classes (such as linear functions), $g(\lambda(\theta))$ is easy to compute but may be a loose upper bound on (2), while more expressive choices lead to tighter relaxation of (2) at the cost of more difficult evaluation (or bounding) of $g(\lambda(\theta))$.

Algorithm 1 Verification with Functional Lagrangians

Input: initial dual parameters $\theta^{(0)}$, learning-rate η, number of iterations T.
for $t = 0, \ldots, T-1$ **do** {`optimization loop`}
  for $k = 0$ **to** K **do** {`potentially in parallel`}
    $d_\theta^{(k)} = \nabla_\theta \left[\max_{x_k} g_k(x_k, \lambda_k, \lambda_{k+1}) \right]$ {`potentially approximate maximization`}
  end for
  $\theta^{(t+1)} = \theta^{(t)} - \eta \sum_{k=0}^K d_\theta^{(k)}$ {`or any gradient based optimization`}
end for
Return: Exact value or guaranteed upper bound on $g(\lambda(\theta^{(T)}))$ {`final evaluation`}

Optimization. With some abuse of notation, for convenience, we write $g_0(x_0, \lambda_0, \lambda_1) := g_0(p_0, \lambda_1)$ and $g_K(x_K, \lambda_K, \lambda_{K+1}) := g_K(p_K, \lambda_K)$, with $\lambda_0 = \lambda_{K+1} = 0$. Then the problem of obtaining the best bound can be written as: $\min_\theta \sum_{k=0}^K \max_{x_k} g_k(x_k, \lambda_k, \lambda_{k+1})$, where the inner maximizations are understood to be performed over the appropriate domains ($\mathcal{P}_0$ for x_0, $\mathcal{X}_k$ for x_k,

$l = 1, \ldots, K-1$ and $\mathcal{P}_K$ for x_K). The overall procedure is described in Algorithm 1: θ is minimized by a gradient-based method in the outer loop; in the inner loop, the decomposed maximization problems over the x_k get solved, potentially in parallel. During optimization, the inner problems can be solved approximately as long as they provide sufficient information about the descent direction for θ.

Guaranteeing the Final Results. For the final verification certificate to be valid, we do require the final evaluation to provide the exact value of $g(\lambda(\theta^{(T)}))$ or an upper bound. In the following section, we provide an overview of novel bounds that we use in our experiments to certify the final results.

3.3 Bounds for Specific Instantiations

The nature of the maximization problems encountered by the optimization algorithm depends on the verification problem as well as the type of chosen Lagrangian multipliers. In some easy cases, like linear multipliers on a ReLU layer, this results in tractable optimization or even closed-form solutions. In other cases however, obtaining a non-trivial upper bound is more challenging. In this section, we detail two such situations for which novel results were required to get tractable bounds: distributionally robust verification and expected softmax-based problems. To the best of our knowledge, these bounds do not appear in the literature and thus constitute a novel contribution.

Distributionally Robust Verification with Linexp Multipliers. We consider the setting where we verify a deterministic network with stochastic inputs and constraints on the input distribution $p_0 \in \mathcal{P}_0$. In particular, we consider $\mathcal{P}_0 = \{\mu + \omega : \omega \sim \mathcal{P}_{noise}\}$, where $\mathcal{P}_{noise}$ denotes a class of zero-mean noise distributions that all satisfy the property of having sub-Gaussian tails (this is true for many common noise distributions including Bernoulli, Gaussian, truncated Gaussian):

$$\text{Sub-Gaussian tail:} \quad \forall i, \forall t \in \mathbb{R},\ \mathbb{E}\left[\exp\left(t\omega_i\right)\right] \leq \exp\left(t^2\sigma^2/2\right).$$

We also assume that each component of the noise ω_i is i.i.d. The functional Lagrangian dual $g(\lambda)$ only depends on the input distribution p_0 via g_0, which evaluates to $g_0(p_0, \lambda_1) = \mathbb{E}_{x\sim p_0}[\lambda_1(x)]$. If we choose λ_1 to be a linear or quadratic function, then $g(\lambda)$ only depends on the first and second moments of p_0. This implies that the verification results will be unnecessarily conservative as they don't use the full information about the distribution p_0. To consider the full distribution it suffices to add an exponential term which evaluates to the moment generating function of the input distribution. Therefore we choose $\lambda_1(x) = \alpha^T x + \exp\left(\gamma^T x + \kappa\right)$ and $\lambda_2(x) = \beta^T x$. The following result then gives a tractable upper bound on the resulting maximization problems:

Proposition 1. *In the setting described above, and with s as the element-wise activation function:*

$$\max_{p_0 \in \mathcal{P}_0} g_0(p_0, \lambda_1) \leq \alpha^T(w\mu + b) + \exp\left(\left\|w^T\gamma\right\|^2 \sigma^2/2 + \gamma^T b + \kappa\right),$$
$$\max_{x \in \mathcal{X}_1} g_1(x, \lambda_1, \lambda_2) \leq \max_{x \in \mathcal{X}_2, z = s(x)} \beta^T(w_2 z + b_2) - \alpha^T x - \exp\left(\gamma^T x + \kappa\right).$$

The maximization in the second equation can be bounded by solving a convex optimization problem (Appendix C.3).

Expected Softmax Problems. Several of the specifications discussed in Section 2.3 (e.g., distributionally robust OOD detection) require us to bound the expected value of a linear function of the softmax. For specifications whose function can be expressed as an expected value: $\boldsymbol{\psi}(p_K) = \mathbb{E}_{x\sim p_K}[c(x)]$, by linearity of the objective w.r.t. the output distribution p_K, the search over the distribution p_k can be simplified to a search over feasible output values x_K:

$$\max_{p_K \in \mathcal{P}_K} \boldsymbol{\psi}(p_K) - \int_{\mathcal{X}_K} \lambda_K(x)\, p_K(x)\, dx = \max_{x \in \mathcal{X}_K} c(x) - \lambda_K(x). \tag{7}$$

Given this observation, the following lets us certify results for linear functions of the $\texttt{softmax}(x)$:

Proposition 2. *For affine λ_K, and $c(x)$ with the following form $c(x) = \mu^T \texttt{softmax}(x)$, $\max_{x\in\mathcal{X}_K} c(x) - \lambda_K(x)$ can be computed in time $O(3^d)$, where $\mathcal{X}_K \subseteq \mathbb{R}^d$.*

We provide a proof of this proposition and a concrete algorithm for computing the solution in Appendix C.2. This setting is particularly important to measure verified confidence and thus to perform robust OOD detection. We further note that while the runtime is exponential in d, d corresponds to the number of labels in classification tasks which is a constant value and does not grow with the size of the network or the inputs to the network. Further, the computation is embarassingly parallel and can be done in $O(1)$ time if 3^d computations can be run in parallel. For classification problems with 10 classes (like CIFAR-10 and MNIST), exploiting this parallelism, we can solve these problems on the order of milliseconds on a cluster of CPUs.

4 Related Work

Verification of Probabilistic Specifications. We recall that in our work, $\mathcal{P}_0$ refers to a space of distributions on the inputs x to a network ϕ, and that we address the following problem: verify that $\forall p \in \mathcal{P}_0,\ \phi(p) \in \mathcal{P}_{out}$, where $\mathcal{P}_{out}$ represents a constraint on the output distribution. In contrast, prior works by Weng et al. [2019], Fazlyab et al. [2019], and Mirman et al. [2021] study probabilistic specifications that involve robustness to probabilistic perturbations of a single input for deterministic networks. This setting can be recovered as a special case within our formalism by letting the class $\mathcal{P}_0$ contain a single distribution p. Conversely, Dvijotham et al. [2018a] study specifications involving stochastic models, but can not handle stochasticity in the input space.

Wicker et al. [2020] define a notion of probabilistic safety for BNNs: $Prob_{w\sim\mathcal{P}_w}[\forall x \in \mathcal{X},\ \phi_w(x) \in \mathcal{C}] \geq p_{\min}$, where ϕ_w denotes the network with parameters w, $\mathcal{P}_w$ denotes the distribution over network weights (e.g., a Gaussian posterior) and $\mathcal{C}$ is a set of safe outputs, and this allows for computation of the probability that a randomly sampled set of weights exhibits safe behaviour. However, in practice, inference for BNNs is carried out by averaging over predictions under the distribution of network weights. In this less restrictive and more practical setting, it suffices if the constraint is satisfied by the probabilistic prediction that averages over sampled weights: $\forall x \in \mathcal{X},\ Prob_{w\sim\mathcal{P}_w}[\phi_w(x) \in \mathcal{C}] \geq p_{\min}$, where $\phi_w(x)$ denotes the distribution over outputs for $x \in \mathcal{X}$. Further, Wicker et al. [2020] also observe that $\min_{x\in\mathcal{X}} Prob_{w\sim\mathcal{P}_w}[\phi_w(x) \in \mathcal{C}] \geq Prob_{w\sim\mathcal{P}_w}[\forall x \in \mathcal{X},\ \phi_w(x) \in \mathcal{C}]$, making the second constraint less restrictive. Cardelli et al. [2019] and Michelmore et al. [2020] consider a similar specification, but unlike the approaches used here and by Wicker et al. [2020], these methods can give statistical confidence bounds but not certified guarantees.

Wicker et al. [2021] improve the classification robustness of Bayesian neural networks by training them to be robust based on an empirical estimate of the average upper bound on the specification violation, for a fixed set of sampled weights. In contrast, our approach provides meaningful guarantees for BNNs trained without considerations to make them more easily verifiable, and the guarantees our framework provides hold for inference based on the true expectation, as opposed to a fixed set of sampled weights.

Our work also generalizes Bitterwolf et al. [2020], which studies specifications of the output distribution of NNs when the inputs and network are deterministic. In contrast, our framework's flexibility allows for stochastic networks as well. Furthermore, while Bitterwolf et al. [2020] are concerned with training networks to be verifiable, their verification method fails for networks trained in a verification-agnostic manner. In our experiments, we provide stronger guarantees for networks that are trained in a verification-agnostic manner.

Lagrangian Duality. Our framework subsumes existing methods that employ Lagrangian duality for NN verification. In Appendix D.1, we show that the functional Lagrangian dual instantiated with linear multipliers is equivalent to the dual from Dvijotham et al. [2018b]. This is also the dual of the LP relaxation [Ehlers, 2017] and the basis for other efficient NN verification algorithms ([Zhang et al., 2018, Singh et al., 2018], for example), as shown in Liu et al. [2021]. For the case of quadratic multipliers and a particular grouping of layers, we show that our framework is equivalent to the Lagrangian dual of the SDP formulation from Raghunathan et al. [2018] (see Appendix D.2).

We also note that similar mathematical ideas on nonlinear Lagrangians have been explored in the optimization literature [Nedich and Ozdaglar, 2008, Feizollahi et al., 2017] but only as a theoretical construct - this has not lead to practical algorithms that exploit the staged structure of optimization problems arising in NN verification. Further, these approaches do not handle stochasticity.

Table 1: Robust OOD Detection: MNIST vs EMNIST (MLP and LeNet) and CIFAR-10 vs CIFAR-100 (VGG-*). BP: Bound-Propagation (baseline); FL: Functional Lagrangian (ours). The reported times correspond to the median of the 500 samples.

OOD Task	Model	#neurons	#params	ϵ	Time (s)		GAUC (%)		AAUC (%)
					BP	FL	BP	FL	
(E)MNIST	MLP	256	2k	0.01	40.1	+38.8	55.4	**65.0**	86.9
				0.03	40.1	+37.4	38.5	**53.1**	88.6
				0.05	38.4	+36.2	18.9	**36.1**	88.8
(E)MNIST	LeNet	0.3M	0.1M	0.01	53.2	+52.4	0.0	**29.8**	71.6
				0.03	52.4	+51.1	0.0	**14.1**	57.6
				0.05	55.4	+54.1	0.0	**3.1**	44.0
CIFAR	VGG-16	3.0M	83k	0.001	50.8	+35.0	0.0	**25.6**	60.9
	VGG-32	5.9M	0.2M	0.001	82.3	+40.9	0.0	**25.8**	64.7
	VGG-64	11.8M	0.5M	0.001	371.7	+48.7	0.0	**29.5**	67.4

5 Experiments

Here, we empirically validate the theoretical flexibility of the framework and its applicability to across several specifications and networks. Crucially, we show that our framework permits verification of probabilistic specifications by effectively handling parameter and input stochasticity across tasks. For all experiments, we consider a layer decomposition of the network such that the intermediate inner problems can be solved in closed form with linear multipliers (See Appendix C.4). We use different bound-propagation algorithms to compute activation bounds based on the task, and generally refer to these methods as BP. Our code is available at `https://github.com/deepmind/jax_verify`.

5.1 Robust OOD Detection on Stochastic Neural Networks

Verification Task. We consider the task of robust OOD detection for stochastic neural networks under bounded ℓ_∞ inputs perturbation with radius ϵ. More specifically, we wish to use a threshold on the softmax value (maximized across labels) to classify whether a sample is OOD. By using verified upper bounds on the softmax value achievable under ϵ-perturbations, we can build a detector that classifies OOD images as such even under ϵ perturbations. We use the Area Under the Curve (AUC) to measure the performance of the detector. Guaranteed AUC (GAUC) is obtained with verified bounds on the softmax, and Adversarial AUC (AAUC) is based on the maximal softmax value found through an adversarial attack. We consider two types of stochastic networks: i) Bayesian neural networks (BNNs) whose posterior distribution is a truncated Gaussian distribution. We re-use the MLP with two hidden layers of 128 units from Wicker et al. [2020] (denoted as MLP) and truncate their Gaussian posterior distribution to three standard deviations, ii) we consider MC-Dropout convolutional neural networks, namely we use LeNet (as in Gal and Ghahramani [2016]) and VGG-style models [Simonyan and Zisserman, 2015].

Method. We use linear Lagrangian multipliers, which gives closed-form solutions for all intermediate inner maximization problems (Appendix C.4). In addition, we leverage Proposition 2 to solve the final inner problem with the softmax specification objective. Further experimental details, including optimization hyper-parameters, are available in Appendix E.1. We compute activation bounds based on an extension of Bunel et al. [2020] to the bilinear case, referred to as BP in Table 1. The corresponding bounds are obtained with probability 1, and if we were to relax these guarantees to only hold up to some probability lower than 1, we note that the method of [Wicker et al., 2020] would offer tighter bounds.

Results. The functional Lagrangian (FL) approach systematically outperforms the Bound-Propagation (BP) baseline. We note that in particular, FL significantly outperforms BP on dropout CNNs, where BP is often unable to obtain any guarantee at all (See Table 1). As the size of the VGG model increases, we can observe that the median runtime of BP increases significantly, while the additional overhead of using FL remains modest.

Table 2: Adversarial Robustness for different BNN architectures trained on MNIST from Wicker et al. [2020]. The accuracy reported for FL and LBP is the % of samples we can certify as robust with probability 1. For each model, we run the experiment for the first 500 test-set samples.

#layers	ϵ	#neurons	LBP Acc. (%)	FL Acc. (%)	LBP Time (s)	FL Time (s)
1	0.025	128	67.0	**77.2**	16.7	+518.3
		256	66.2	**76.4**	16.1	+522.8
		512	60.2	**74.6**	16.0	+522.4
2	0.001	256	57.0	**70.0**	16.8	+516.5
		512	79.6	**87.4**	17.0	+517.3
		1024	39.4	**42.4**	17.1	+514.1

5.2 Adversarial Robustness for Stochastic Neural Networks

Verification Task. For this task, we re-use the BNNs trained on MNIST [LeCun and Cortes, 2010] from Wicker et al. [2020] (with the Gaussian posterior truncated to three standard deviations bounded). We use the same setting as Wicker et al. [2020] and study the classification robustness under ℓ_∞ perturbations for 1-layered BNNs and 2-layered BNN at radii of $\epsilon = 0.025$ and $\epsilon = 0.001$ respectively. We recall, as mentioned earlier in Section 4, that the specification we study is different from that studied in [Wicker et al., 2020].

Method We use the same solving methodology as in Section 5.1. To compute bounds on the activations, we use the bilinear LBP method proposed in Wicker et al. [2020].

Results. Across settings (Table 2) our approach is able to significantly improve on the guarantees provided by the LBP baseline, while also noting that our method incurs an increased compute cost.

5.3 Distributionally Robust OOD Detection

Verification Task. We bound the largest softmax probability across all labels for OOD inputs over noisy perturbations of the input where the noise is drawn from a family of distributions with only two constraints for each $p \in \mathcal{P}_{noise}$: $\omega \in [-\epsilon, \epsilon]$ with prob. $1, \mathbb{E}_{\omega \sim p}\left[\exp\left(\omega t\right)\right] \leq \exp\left(\sigma^2 t^2/2\right)$, for given constraints $\epsilon, \sigma > 0$. The first constraint corresponds to a restriction on the support of the noise distribution, and the second constraint requires that the noise distribution is sub-Gaussian with parameter σ. We note that no prior verification method, to the best of our knowledge, addresses this setting. Thus, as a baseline, we use methods that only use the support of the distribution and perform verification with respect to worst-case noise realizations within these bounds.

Method. We use a 3-layer CNN trained on MNIST with the approach from Hein et al. [2019] (details in Appendix E.3), and use functional multipliers of the form: $\lambda_k\left(x\right) = \theta_k^T x$ for $k > 1$ and linear-exponential multipliers for the input layer: $\lambda_1\left(x\right) = \theta_1^T x + \exp\left(\gamma_1^T x + \kappa_1\right)$ (method denoted by FL-LinExp). As baselines, we consider a functional Lagrangian setting with linear multipliers that only uses information about the expectation of the noise distribution (method denoted by FL-Lin), and a BP baseline that only uses information about the bounds on the support of the noise distribution $(-\epsilon, \epsilon)$ (activation bounds are computed using Bunel et al. [2020]). The inner maximization of g_k for $K - 1 \geq k \geq 2$ can be done in closed form and we use approaches described in Propositions 1 and 2 to respectively solve $\max g_0, \max g_1$ and $\max g_K$. We use $\epsilon = 0.04, \sigma = 0.1$.

Table 3: Guaranteed Area Under Curve (GAUC) values in a distributionally robust setting. The stochastic formulation of the Functional Lagrangian with Linear-Exponential (LinExp) multipliers gets the highest guaranteed AUC.

Model	#neurons	GAUC (%)			Timing (s)		
		BP	FL-Lin	FL-LinExp	BP	FL-Lin	FL-LinExp
CNN	9972	5.4	5.6	**23.2**	227.7	+661.8	+760.7

Results. FL-LinExp achieves the highest guaranteed AUC (See Table 3), showing the value of accounting for the noise distribution, instead of relying only on bounds on the input noise.

6 Conclusion

We have presented a general framework for verifying probabilistic specifications, and shown significant improvements upon existing methods, even for simple choices of the Lagrange multipliers where we can leverage efficient optimization algorithms. We believe that our approach can be significantly extended by finding new choices of multipliers that capture interesting properties of the verification problem while leading to tractable optimization problems. This could lead to discovery of new verification algorithms, and thus constitutes an exciting direction for future work.

Limitations. We point out two limitations in the approach suggested by this work. First, the guarantees provided by our approach heavily depend on the bounding method used to obtain the intermediate bounds – this is consistent with prior work on verifying deterministic networks [Dathathri et al., 2020, Wang et al., 2021], where tighter bounds result in much stronger guarantees. Second, as noted in Section 3.1, our approach can only handle probability distributions that have bounded support, and alleviating this assumption would require further work.

Broader Impact and Risks. Our work aims to improve the reliability of neural networks in settings where either the model or its inputs exhibit probabilistic behaviour. In this context, we anticipate this work to be largely beneficial and do not envision malicious usage. However, the guarantees of our method crucially depend on having an accurate modeling of the uncertainty. Failing that can result in an overestimation of the reliability of the system, which can have catastrophic consequences in safety-critical scenarios. In this regard, we advocate special care in the design of the specification when applying our approach to real-world use cases.

References

Maksym Andriushchenko, Francesco Croce, Nicolas Flammarion, and Matthias Hein. Square attack: a query-efficient black-box adversarial attack via random search. *European Conference on Computer Vision*, 2020.

Dimitri P Bertsekas. *Convex optimization algorithms*. Athena Scientific Belmont, 2015.

Julian Bitterwolf, Alexander Meinke, and Matthias Hein. Provable worst case guarantees for the detection of out-of-distribution data. *Advances in Neural Information Processing Systems*, 2020.

Rudy Bunel, Oliver Hinder, Srinadh Bhojanapalli, et al. An efficient nonconvex reformulation of stagewise convex optimization problems. *Advances in Neural Information Processing Systems*, 2020.

Luca Cardelli, Marta Kwiatkowska, Luca Laurenti, Nicola Paoletti, Andrea Patane, and Matthew Wicker. Statistical guarantees for the robustness of bayesian neural networks. *International Joint Conference on Artificial Intelligence*, 2019.

Sumanth Dathathri, Krishnamurthy Dvijotham, Alexey Kurakin, Aditi Raghunathan, Jonathan Uesato, Rudy Bunel, Shreya Shankar, Jacob Steinhardt, Ian Goodfellow, Percy Liang, et al. Enabling certification of verification-agnostic networks via memory-efficient semidefinite programming. *Advances in Neural Information Processing Systems*, 2020.

Krishnamurthy Dvijotham, Marta Garnelo, Alhussein Fawzi, and Pushmeet Kohli. Verification of deep probabilistic models. *NeurIPS 2018 Workshop on Security in Machine Learning*, 2018a.

Krishnamurthy Dvijotham, Robert Stanforth, Sven Gowal, Timothy A Mann, and Pushmeet Kohli. A dual approach to scalable verification of deep networks. *Conference on Uncertainty in Artificial Intelligence*, 2018b.

Ruediger Ehlers. Formal verification of piece-wise linear feed-forward neural networks. *International Symposium on Automated Technology for Verification and Analysis*, 2017.

Mahyar Fazlyab, Manfred Morari, and George J Pappas. Probabilistic verification and reachability analysis of neural networks via semidefinite programming. *Conference on Decision and Control (CDC)*, 2019.

Mohammad Javad Feizollahi, Shabbir Ahmed, and Andy Sun. Exact augmented lagrangian duality for mixed integer linear programming. *Mathematical Programming*, 2017.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. *International Conference on Machine Learning*, 2016.

Robert Grone, Charles R Johnson, Eduardo M Sá, and Henry Wolkowicz. Positive definite completions of partial hermitian matrices. *Linear algebra and its applications*, 1984.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. *Computer Vision and Pattern Recognition*, 2016.

Matthias Hein, Maksym Andriushchenko, and Julian Bitterwolf. Why relu networks yield high-confidence predictions far away from the training data and how to mitigate the problem. *Conference on Computer Vision and Pattern Recognition*, 2019.

Guy Katz, Clark Barrett, David L Dill, Kyle Julian, and Mykel J Kochenderfer. Reluplex: An efficient smt solver for verifying deep neural networks. *International Conference on Computer Aided Verification*, 2017.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 2015.

Yann LeCun and Corinna Cortes. MNIST handwritten digit database. *NIST*, 2010.

Changliu Liu, Tomer Arnon, Christopher Lazarus, Clark Barrett, and Mykel J Kochenderfer. Algorithms for verifying deep neural networks. *Foundations and Trends in Optimization*, 2021.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *International Conference on Learning Representations*, 2017.

Rhiannon Michelmore, Matthew Wicker, Luca Laurenti, Luca Cardelli, Yarin Gal, and Marta Kwiatkowska. Uncertainty quantification with statistical guarantees in end-to-end autonomous driving control. *International Conference on Robotics and Automation (ICRA)*, 2020.

Matthew Mirman, Timon Gehr, and Martin Vechev. Robustness certification of generative models. *International Conference on Programming Language Design and Implementation*, 2021.

Angelia Nedich and Asuman Ozdaglar. A geometric framework for nonconvex optimization duality using augmented lagrangian functions. *Journal of global optimization*, 2008.

Aditi Raghunathan, Jacob Steinhardt, and Percy S Liang. Semidefinite relaxations for certifying robustness to adversarial examples. *Advances in Neural Information Processing Systems*, 2018.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *International Conference on Learning Representations*, 2015.

Gagandeep Singh, Timon Gehr, Matthew Mirman, Markus Püschel, and Martin T Vechev. Fast and effective robustness certification. *Advances in Neural Information Processing Systems*, 2018.

Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.

Lieven Vandenberghe and Martin S Andersen. Chordal graphs and semidefinite optimization. *Foundations and Trends in Optimization*, 2015.

Shiqi Wang, Huan Zhang, Kaidi Xu, Xue Lin, Suman Jana, Cho-Jui Hsieh, and J. Zico Kolter. Beta-crown: Efficient bound propagation with per-neuron split constraints for complete and incomplete neural network verification. *ICML 2021 Workshop on Adversarial Machine Learning*, 2021.

Lily Weng, Pin-Yu Chen, Lam Nguyen, Mark Squillante, Akhilan Boopathy, Ivan Oseledets, and Luca Daniel. Proven: Verifying robustness of neural networks with a probabilistic approach. *International Conference on Machine Learning*, 2019.

Matthew Wicker, Luca Laurenti, Andrea Patane, and Marta Kwiatkowska. Probabilistic safety for bayesian neural networks. *Conference on Uncertainty in Artificial Intelligence*, 2020.

Matthew Wicker, Luca Laurenti, Andrea Patane, Zhoutong Chen, Zheng Zhang, and Marta Kwiatkowska. Bayesian inference with certifiable adversarial robustness. *International Conference on Artificial Intelligence and Statistics*, 2021.

Dong Yu and Li Deng. *Automatic Speech Recognition: A Deep Learning Approach*. Springer, 2014.

Huan Zhang, Tsui-Wei Weng, Pin-Yu Chen, Cho-Jui Hsieh, and Luca Daniel. Efficient neural network robustness certification with general activation functions. *Advances in Neural Information Processing Systems*, 2018.

Appendix

A Probabilistic Specifications: Examples

Below we provide further examples of specifications that can be captured by our framework.

Uncertainty calibration. Another desirable specification towards ensuring reliable uncertainty calibration for NNs is that the expected uncertainty in the predictions increases monotonically with an increase in the variance of the input-noise distribution. Formally, consider a set of zero-mean distributions $\mathcal{P}_{noise}$ with diagonal covariance matrices. For any two such noise distributions $p_{\omega_1}, p_{\omega_2} \in \mathcal{P}_{noise}$ such that $\mathrm{Var}(p_{\omega_1}) \succcurlyeq \mathrm{Var}(p_{\omega_2})$ (where $\succcurlyeq$ is the element-wise inequality and Var denotes the diagonal of the covariance matrix) and a given image x from the training distribution, we wish to guarantee that the expected entropy of the resulting predictions corresponding to p_{ω_1} is greater than that of p_{ω_2}, i.e., $\mathbb{E}_{x_1 \sim x + p_{\omega_1}}\left[H(\texttt{softmax}(\phi(x_1)\right] \geq \mathbb{E}_{x_2 \sim x + p_{\omega_2}}\left[H(\texttt{softmax}(\phi(x_2)\right]$, where H is the entropy functional: $H(p) = -\sum_{i=1}^{|\mathcal{Y}|} p_i \log p_i$. Intuitively, this captures the desired behaviour that as the strength of the noise grows, the expected uncertainty in the network predictions increases. We can capture this specification within the formulation described by equation 1, by letting:

1. $\mathcal{P}_0 = \mathcal{P}_{noise} \times \mathcal{P}_{noise}$,
2. For a given NN ϕ and an input x, we define another network $\bar{\phi} : \mathcal{X} \times \mathcal{X} \to \mathcal{P}(\mathcal{Y}) \times \mathcal{P}(\mathcal{Y})$, where $\bar{\phi}$ is such that: $\bar{\phi}(a, b) = (\phi(x + a), \phi(x + b))$.

We can then define the verification problem as certifying that the following holds :

$$\boldsymbol{\psi}\left(\bar{\phi}(\bar{p}_0, \bar{\bar{p}}_0)\right) \coloneqq -\left(\mathop{\mathbb{E}}_{(y_1,y_2)\sim\bar{\phi}(\bar{p}_0,\bar{\bar{p}}_0)}\left[H\left(\texttt{softmax}\left(y_1\right)\right) - H\left(\texttt{softmax}\left(y_2\right)\right)\right]\right) \leq 0,$$
$$\forall\left(\bar{p}_0, \bar{\bar{p}}_0\right) \in \mathcal{P}_0 \text{ such that } \mathrm{Var}(\bar{p}_0) \succcurlyeq \mathrm{Var}(\bar{\bar{p}}_0).$$

Robust VAE In Dathathri et al. [2020], the authors consider a specification that corresponds to certifying low reconstruction losses for a VAE decoder over a set of inputs in the neighborhood of the latent-variable mean predicted by the encoder for a given image. A natural generalization of this specification is one where low reconstruction error is guaranteed in expectation, since in practice the latent-representations that are fed into the decoder are drawn from a normal distribution whose mean and variance are predicted by the encoder. A more general specification is one where we wish to verify that for a set of norm-bounded points around a given input, the expected reconstruction error from the VAE is small. Formally, for a VAE ϕ (note that a VAE directly fits within our framework, where the distribution for the latent-variable can be obtained as the output of a stochastic layer), a given threshold $\tau \in \mathbb{R}_+$ and for a set of inputs $\mathcal{S}$, we wish to certify that the following holds:

$$\forall p_0 \in \mathcal{P}_0, \quad \boldsymbol{\psi}(\phi(p_0)) \coloneqq \mathop{\mathbb{E}}_{\mu_\phi^{s'} \sim \phi(p_0)}\left[\| \mathop{\mathbb{E}}_{s\sim p_0}[s] - \mu_\phi^{s'} \|_2^2\right] - \tau \leq 0; \tag{8}$$

where $\mathcal{P}_0 = \{\delta_s : s \in \mathcal{S}\}$.

B Proof of Functional Lagrangian Theorem

Proof of Theorem 1

Proof. The optimization problem (2) is equivalent to the following optimization problem:

$$\max_{p_0, p_1, \ldots, p_K} \quad \boldsymbol{\psi}\left(p_K\right) \tag{9a}$$

$$\text{Subject to } p_{k+1}\left(y\right) = \int \pi_k\left(y|x\right) p_k\left(x\right) dx \quad \forall y \in \mathcal{X}_{k+1},\ \forall k \in \{0, \ldots, K-1\}$$

$$p_0 \in \mathcal{P}_0 \tag{9b}$$

where $\boldsymbol{\psi}$ is a functional of the output distribution p_K. We refer to the space of probability measures on $\mathcal{X}_k$ as $\mathcal{P}_k$ (not that for $k = 0$, this may not be the whole space of probability measures but a

constrained set of measures depending on the specification we would like to verify). The dual of this optimization problem can be written as follows:

$$\max_{p_0 \in \mathcal{P}_0, p_1 \in \mathcal{P}_1, \ldots} \boldsymbol{\psi}\left(p_K\right) - \sum_{k=0}^{K-1} \int \lambda_{k+1}\left(z\right) p_{k+1}\left(z\right) dz + \sum_{k=0}^{K-1} \int_{\mathcal{X}_{k+1}, \mathcal{X}_k} \lambda_{k+1}\left(z\right) \pi_k\left(z|x\right) p_k\left(x\right) dx dz,$$

where we assigned Lagrange multipliers $\lambda_{k+1}\left(y\right)$ for every $y \in \mathcal{X}_{k+1}$. The above optimization problem can be decomposed as follows:

$$\boldsymbol{\psi}\left(p_K\right) - \int_{\mathcal{X}_K} \lambda_K\left(x\right) p_K\left(x\right) dx + \sum_{k=0}^{K-1} \int_{\mathcal{X}_k} \left(\int_{\mathcal{X}_{k+1}} \lambda_{k+1}\left(y\right) \pi_k\left(y|x\right) dy - \lambda_k\left(x\right) \right) p_k\left(x\right) dx.$$

This is a separable problem in each p_k and since p_k is constrained to be a probability measure, the optimal choice of p_k (for $k = 1, \ldots, K-1$) is a δ measure with probability 1 assigned to the the $x \in \mathcal{X}_k$ that maximizes:

$$\int_{\mathcal{X}_{k+1}} \lambda_{k+1}\left(y\right) \pi_k\left(y|x\right) dy - \lambda_k\left(x\right).$$

The optimization over p_0 can be rewritten as follows:

$$\max_{p_0 \in \mathcal{P}_0} \int_{\mathcal{X}_0} \left(\int_{\mathcal{X}_1} \lambda_1\left(y\right) \pi_0\left(y|x\right) dy \right) p_0\left(x\right) dx.$$

The optimization over p_K can be rewritten as follows:

$$\boldsymbol{\psi}^\star\left(\lambda_K\right) = \max_{p_K \in \mathcal{P}_K} \boldsymbol{\psi}\left(p_K\right) - \int \lambda_K\left(x\right) p_K\left(x\right) dx.$$

The overall dual problem can be rewritten as:

$$\boldsymbol{\psi}^\star\left(\lambda_K\right) + \sum_{k=1}^{K-1} \max_{x \in \mathcal{X}_k} \left(\int_{\mathcal{X}_{k+1}} \lambda_{k+1}\left(y\right) \pi_k\left(y|x\right) dy - \lambda_k\left(x\right) \right) + \max_{p_0 \in \mathcal{P}_0} \int_{\mathcal{X}_0} \left(\int_{\mathcal{X}_1} \lambda_1\left(y\right) \pi_0\left(y|x\right) dy \right) p_0\left(x\right) dx.$$

Writing this in terms of expected values, we obtain $g\left(\lambda\right)$. Plugging in $\lambda^\star$ into $g\left(\lambda\right)$, all the terms cancel except the first term which evaluates to:

$$\max_{p_0 \in \mathcal{P}_0} \mathop{\mathbb{E}}_{x \sim p_0} \left[\mathop{\mathbb{E}}_{y \in \pi_0(x)} \left[\lambda_1^\star\left(x\right) \right] \right] = \max_{p_0 \in \mathcal{P}_0} \mathop{\mathbb{E}}_{x \sim p_0} \left[\mathop{\mathbb{E}}_{y \in \pi_0(x)} \left[\mathop{\mathbb{E}}_{z \sim \pi_1(x)} \left[\lambda_2^\star\left(z\right) \right] \right] \right] \ldots = \max_{p_0 \in \mathcal{P}_0} \boldsymbol{\psi}\left(p_0\right)$$

□

C Additional Theoretical Results

C.1 Computation of Expected Values

Since $\pi_k\left(x\right)$ is typically a distribution that one can sample from easily (as it is required to perform forward inference through the neural network), estimating this expectation via sampling is a viable option. However, in order to turn this into verified bounds on the specification, one needs to appeal to concentration inequalities and the final guarantees would only be probabilistically valid. We leave this direction for future work.

Instead, we focus on situations where the expectations can be computed in closed form. In particular, we consider layers of the form $\pi_k(x) = ws(x) + b, (w, b) \sim p_k^w$, where s is an element-wise function like ReLU, sigmoid or tanh and (w, b) represents a fully connected or convolutional layer. We consider a general form of Lagrange multipliers as a sum of quadratic and exponential terms as follows:

$$\lambda(x) = q^T x + \frac{1}{2} x^T Q x + \kappa \exp\left(\gamma^T x\right).$$

Let:

$$\tilde{s}(x) = \begin{pmatrix} 1 \\ s(x) \end{pmatrix}, \tilde{Q} = \begin{pmatrix} 0 & q^T \\ q & Q \end{pmatrix}, \tilde{w} = \begin{pmatrix} b & w \end{pmatrix}.$$

Then:

$$\lambda(Ws(x) + b) = \frac{1}{2} (\tilde{s}(x))^T \tilde{w}^T \tilde{Q} \tilde{w} \tilde{s}(x) + \kappa \exp\left(\gamma^T \tilde{w} \tilde{s}(x)\right).$$

Taking expected values with respect to $\tilde{W} \sim p_k^w$, we obtain:

$$\mathbb{E}[\lambda(ws(x) + b)] = \frac{1}{2} (\tilde{s}(x))^T \mathop{\mathbb{E}}_{\tilde{s}(x)}\left[\tilde{w}^T \tilde{Q} \tilde{w}\right] + \kappa \prod_{i,j} \mathbb{E}[\exp(\gamma_i \tilde{w}_{ij} \tilde{s}(x_j))], \tag{10}$$

where we have assumed that each element of $\tilde{W}_{ij}$ is independently distributed. The first term in (10) evaluates to:

$$\frac{1}{2}\text{Trace}\left(\text{Cov}(\tilde{w}\tilde{s}(x))\tilde{Q}\right) + \frac{1}{2}\left(\mathbb{E}[\tilde{w}\tilde{s}(x)]\right)^\top \tilde{Q}\left(\mathbb{E}[\tilde{w}\tilde{s}(x)]\right),$$

and the second one to:

$$\kappa \prod_{i,j} \text{mgf}_{ij}(\gamma_i \tilde{s}_j(x)),$$

where $\text{Cov}(X)$ refers to the covariance matrix of the random vector, and mgf_{ij} refers to the moment generating function of the random variable $\tilde{w}_{ij}$:

$$\text{mgf}_{ij}(\theta) = \mathbb{E}[\exp(\tilde{w}_{ij}\theta)].$$

The details of this computation for various distributions on $\tilde{w}$ (Gaussian posterior, MC-dropout) are worked out below.

Diagonal Gaussian posterior. Consider a BNN with a Gaussian posterior, $\tilde{w} \sim \mathcal{N}\left(\mu, \text{diag}\left(\sigma^2\right)\right)$, where $\mu, \sigma \in \mathbb{R}^{mn}$, let $\text{Mat}(\mu) \in \mathbb{R}^{m \times n}$ denote a reshaped version of μ. Then, we have:

$$\begin{aligned}\mathbb{E}[\lambda(ws(x) + b)] = &\frac{1}{2}\text{Trace}\left(\text{diag}\left(\text{Mat}\left(\sigma^2\right)\tilde{s}(x)\right)\tilde{Q}\right) + \frac{1}{2}(\text{Mat}(\mu)\tilde{s}(x))^T \tilde{Q}(\text{Mat}(\mu)\tilde{s}(x)) \\ &+ \kappa \prod_{i,j} \exp\left(\text{Mat}(\mu)_{ij}\tilde{s}(x_j)\gamma_i + \frac{1}{2}\text{Mat}\left(\sigma^2\right)_{ij}(\tilde{s}(x_j)\gamma_i)^2\right).\end{aligned}$$

MC dropout. Now assume a neural network with dropout: $\tilde{w} = \mu \odot \text{Bernoulli}(p_{\text{dropout}})$, where $\mu \in \mathbb{R}^{mn}$ denotes the weight in the absence of dropout and $p_{\text{dropout}} \in \mathbb{R}^{mn}$ denotes the probability of dropout. Then, we have:

$$\begin{aligned}\mathbb{E}[\lambda(ws(x) + b)] = &\frac{1}{2}\text{Trace}\left(\text{diag}\left(\text{Mat}\left(\mu \odot p_{\text{dropout}} \odot (1 - p_{\text{dropout}})\right)\tilde{s}(x)\right)\tilde{Q}\right) \\ &+ \frac{1}{2}(\text{Mat}(\mu \odot p_{\text{dropout}})\tilde{s}(x))^T \tilde{Q}(\text{Mat}(\mu \odot p_{\text{dropout}})\tilde{s}(x)) \\ &+ \kappa \prod_{i,j}\left(\text{Mat}(p_{\text{dropout}})_{ij} \exp\left(\text{Mat}(\mu)_{ij}\tilde{s}(x_j)\gamma_i\right) + 1 - \text{Mat}(p_{\text{dropout}})_{ij}\right).\end{aligned}$$

C.2 Expected-Softmax Optimization

We describe an algorithm to solve optimization problems of the form

$$\max_{\ell \le x \le u} \mu^T \texttt{softmax}(x) - \lambda^T x$$

Our results will rely on the following lemma:

Proposition 3. *Consider the function*

$$f(x) = \frac{\sum_i \mu_i \exp(x_i) + D}{\sum_j \exp(x_j) + B} - \lambda^T x$$

where $B \geq 0$ and $D = 0$ if $B = 0$. Let $r = \frac{D}{B}$ if $B > 0$ and 0 otherwise. Define the set

$$\Delta = \left\{ \kappa \in \mathbb{R} : (\kappa - r) \left(\prod_{i=1}^{n} (\mu_i - \kappa) \right) - \sum_{i=1}^{n} \mu_i (1-r) \lambda_i \left(\prod_{j \neq i} (\mu_j - \kappa) \right) = 0 \right\}$$

which is a set with at most $n+1$ elements. Define further

$$\Delta_f = \begin{cases} \left\{ \kappa \in \Delta : 0 < \frac{\lambda}{\mu - \kappa} < 1, \sum_{i=1}^{n} \frac{\lambda_i}{\mu_i - \kappa} \leq 1 \right\} & \text{if } B > 0 \\ \left\{ \kappa \in \Delta : 0 < \frac{\lambda}{\mu - \kappa} < 1, \sum_{i=1}^{n} \frac{\lambda_i}{\mu_i - \kappa} = 1 \right\} & \text{if } B = 0 \end{cases}$$

Then, the set of stationary points of f are given by

$$\left\{ \log \left(h \left(\frac{\lambda}{\mu - \kappa} \right) \right) : \kappa \in \Delta_f \right\}$$

where

$$h(v) = \begin{cases} \frac{Bv}{1 - \mathbf{1}^T v} \text{ if } B > 0 \\ \{\theta v : \theta > 0\} \text{ if } B = 0 \end{cases}$$

Proof. Differentiating with respect to x_i, we obtain

$$\frac{\exp(x_i)}{\sum_j \exp(x_j) + B} \left(\mu_i - \left(\frac{\sum_j \mu_j \exp(x_j) + D}{\sum_j \exp(x_j) + B} \right) \right) - \lambda_i = p_i \left(\mu_i - \mu^T p - q \right) - \lambda_i$$

where

$$p_i = \frac{\exp(x_i)}{\sum_j \exp(x_j) + B}, q = \frac{D}{\sum_j \exp(x_j) + B}.$$

If we set the derivative to 0 (to obtain a stationary point) we obtain the following coupled set of equations in p, q, κ:

$$\begin{aligned} p_i &= \frac{\lambda_i}{\mu_i - \kappa} \quad i = 1, \ldots, n \\ q &= r \left(1 - \sum_i p_i \right), \\ \kappa &= \sum_i \mu_i p_i + q, \end{aligned}$$

where $r = \frac{D}{B}$. We can solve this by first solving the scalar equation

$$\kappa - r = \sum_i \frac{\mu_i (1-r) \lambda_i}{\mu_i - \kappa}$$

for κ (this is derived by adding up the first n equations above weighted by μ_i and plugging in the value of q). This equation can be converted into a polynomial equation in κ

$$(\kappa - r) \left(\prod_i (\mu_i - \kappa) \right) - \sum_i \mu_i (1-r) \lambda_i \left(\prod_{j \neq i} (\mu_j - \kappa) \right) = 0$$

which we can solve for all possible real solutions, denote this set Δ. Note that this set has at most $n+1$ elements since it is the set of real solutions to a degree $n+1$ polynomial.

In order to recover x from this, we first recall:

$$p_i = \frac{\lambda_i}{\mu_i - \kappa}$$

Since $p_i = \frac{\exp x_i}{\sum_j \exp(x_j)+B}$, we require that $p_i \in (0,1)$ and $\sum_i p_i \leq 1$ (with equality when $B = 0$ and strict inequality when $B = 1$. We thus filter Δ to the set of κ that lead to p satisfying these properties to obtain Δ_f.

Once we have these, we are guaranteed that for each $\kappa \in \Delta_f$, we can define p_i as above and solve for x_i by solving the linear system of equations

$$u_i = p_i \left(\sum_j u_j + B \right) \quad i = 1, 2, \ldots, n$$

which can be solved as:

$$u = B \left(I - p\mathbf{1}^T \right)^{-1} p = \frac{Bp}{1 - \mathbf{1}^T p}, x = \log(u)$$

if $B > 0$ since the matrix $I - p\mathbf{1}^T$ is strictly diagonally dominant and hence invertible, and we applied the Woodbury identity to compute the explicit inverse.

If $B = 0$, we have $p = \texttt{softmax}(x)$ and can recover x as

$$x = \log(p\theta)$$

for any $\theta > 0$. □

The above lemma allows us to characterize all stationary points of the function

$$\mu^T \texttt{softmax}(x) - \lambda^T x$$

when a subset of entries of x are fixed to their upper or lower bounds, and we search for stationary points wrt the remaining free variables. Given this ability, we can develop an algorithm to globally optimize $\mu^T \texttt{softmax}(x) - \lambda^T x$ subject to bound constraints by iterating over all possible 3^n configurations of binding constraints (each variable could be at its lower bound, upper bound or strictly between them). In this way, we are guaranteed to loop over all local optima, and by picking the one achieving the best objective value, we can guarantee that we have obtained the global optimum. The overall algorithm is presented in Algorithm 2.

Proposition 4. *Algorithm 2 finds the global optimum of the optimization problem*

$$\min_{x:\ell \leq x \leq u} \mu^T \texttt{softmax}(x) - \lambda^T x$$

and runs in time $O(n3^n)$ where n is the dimension of x.

Algorithm 2 Solving softmax layer problem via exhaustive enumeration

Inputs: $\lambda, \mu, \ell, u \in \mathbb{R}^n$
$x^\star \leftarrow \ell$
$f_{opt}(x) \leftarrow \mu^T \texttt{softmax}(x) - \lambda^T x$, $f^\star \leftarrow f_{opt}(x^\star)$
for $v \in [\text{Lower}, \text{Upper}, \text{Interior}]^n$ **do**
  nonbinding$[i] \leftarrow (v[i] == \text{Interior})$, $x_i \leftarrow \begin{cases} l[i] \text{ if } v[i] = \text{Lower} \\ u[i] \text{ if } v[i] = \text{Upper} \end{cases}$ for $i = 1, \ldots, n$
  $B \leftarrow \sum_{i \text{ such that nonbinding}[i]==\text{False}} \exp(x[i])$
  $D \leftarrow \sum_{i \text{ such that nonbinding}[i]==\text{False}} \mu[i] \exp(x[i])$
  Use proposition 3 to find the set of stationary points $\mathcal{S}_f$ of the function

$$f(x) = \frac{\sum_{i \text{ such that nonbinding}[i]} \mu_i \exp(x_i) + D}{\sum_{j \text{ such that nonbinding}[j]} \exp(x_j) + B} - \sum_{j \text{ such that nonbinding}[j]} \lambda[j] x_j$$

  for $x^s \in \mathcal{S}_f$ **do**
    if $x_i^s \in [\ell[i], u[i]] \quad \forall i$ s.t nonbinding$[i]$ **then**
      $x_i \leftarrow x_i^s \quad \forall i$ s.t nonbinding$[i]$.
      if $f_{opt}(x) > f^\star$ **then**
        $x^\star \leftarrow x$
        $f^\star \leftarrow f_{opt}(x^\star)$
      end if
    end if
  end for
end for
Return $x^\star, f^\star$

C.3 Input Layer with Linear-Exponential Multipliers

We recall the setting from Proposition 1. Let $\lambda_1(x) = \alpha^T x + \exp\left(\gamma^T x + \kappa\right)$, $\lambda_2(x) = \beta^T x$, $g_0^\star = \max_{p_0 \in \mathcal{P}_0} g_0(p_0, \lambda_1)$, and $g_1^\star = \max_{x \in \mathcal{X}_1} g_0(x, \lambda_1, \lambda_2)$.

Proposition 5. *In the setting described above, and with s as the element-wise activation function:*

$$g_0^\star \leq \alpha^T (w\mu + b) + \exp\left(\frac{\left\|w^T\gamma\right\|^2 \sigma^2}{2} + \gamma^T b + \kappa\right),$$
$$g_1^\star \leq \max_{\substack{x \in \mathcal{X}_2 \\ z = s(x)}} \beta^T (w_2 z + b_2) - \alpha^T x - \exp\left(\gamma^T x + \kappa\right).$$

The maximization in the second equation can be bounded by solving the following convex optimization problem:

$$\min_{\eta \in \mathbb{R}^n, \zeta \in \mathbb{R}_+} \zeta(\log(\zeta) - 1 - \kappa) + \mathbf{1}^T \max\left(\left(\eta + w_2^T \beta\right) \odot s(l_2), \left(\eta + w_2^T \beta\right) \odot s(u_2)\right) + \sum_i s^\star(\alpha_i + \zeta\gamma_i, \eta_i, l_{2i}, u_{2i}),$$

where $s^\star(a, b, c, d) = \max_{z \in [c,d]} -az - bs(z)$.

Proof.

$$
\begin{aligned}
&\max_{\substack{x\in\mathcal{X}_2\\ z=s(x)}} \beta^T\left(w_2 z+b_2\right)-\alpha^T x-\exp\left(\alpha^T x+\kappa\right),\\
&\quad\leq \min_{\eta}\max_{x\in\mathcal{X}_2, z\in s(\mathcal{X}_2)} \eta^T\left(z-s\left(x\right)\right)+\beta^T\left(w_2 z+b_2\right)-\alpha^T x-\exp\left(\gamma^T x+\kappa\right),\\
&\quad\leq \min_{\eta,\zeta}\max_{x\in\mathcal{X}_2,t} \mathbf{1}^T\max\left(\left(\eta+w_2^T\beta\right)\odot s\left(l_2\right),\left(\eta+w_2^T\beta\right)\odot s\left(u_2\right)\right)+\beta^T b_2-\alpha^T x\\
&\qquad-\eta^T s\left(x\right)-\exp\left(t\right)+\zeta\left(t-\gamma^T x-\kappa\right),\\
&\quad\leq \min_{\eta,\zeta}\zeta\left(\log\left(\zeta\right)-1-\kappa\right)+\mathbf{1}^T\max\left(\left(\eta+w_2^T\beta\right)\odot s\left(l_2\right),\left(\eta+w_2^T\beta\right)\odot s\left(u_2\right)\right)\\
&\qquad+\max_{l_2\leq x\leq u_2}-\left(\alpha+\zeta\gamma\right)^T x-\eta^T s\left(x\right),\\
&\quad\leq \min_{\eta,\zeta}\zeta\left(\log\left(\zeta\right)-1-\kappa\right)+\mathbf{1}^T\max\left(\left(\eta+w_2^T\beta\right)\odot s\left(l_2\right),\left(\eta+w_2^T\beta\right)\odot s\left(u_2\right)\right)\\
&\qquad+\sum_i \max_{l_{2i}\leq x_i\leq u_{2i}}-\left(\alpha_i+\zeta\gamma_i\right)x_i-\eta_i s\left(x_i\right).
\end{aligned}
$$

□

C.4 Inner Problem with Linear Multipliers

In its general form, the objective function of the inner maximization problem can be expressed as:

$$g_k(x_k,\lambda_k,\lambda_{k+1})=\mathop{\mathbb{E}}_{y\sim\pi_k(x_k)}\left[\lambda_{k+1}\left(y\right)\right]-\lambda_k\left(x_k\right). \tag{11}$$

We assume that the layer is in the form $y=Ws\left(x\right)+b$, where W and b are random variables and s is an element-wise activation function. Then we can rewrite $g_k(x_k,\lambda_k,\lambda_{k+1})$ as:

$$g_k(x_k,\lambda_k,\lambda_{k+1})=\mathop{\mathbb{E}}_{W,b}\left[\lambda_{k+1}\left(W\max\{x_k,0\}+b\right)\right]-\lambda_k\left(x_k\right). \tag{12}$$

We now use the assumption that λ_{k+1} is linear: $\lambda_{k+1}:y\mapsto\theta_{k+1}^\top y$. Then the problem can equivalently be written as:

$$
\begin{aligned}
g_k(x_k,\lambda_k,\lambda_{k+1})&=\mathop{\mathbb{E}}_{W,b}\left[\theta_{k+1}^\top\left(Ws\left(x_k\right)+b\right)\right]-\theta_k^T x_k,\\
&=\left(\mathbb{E}\left[W\right]^\top\theta_{k+1}\right)^\top s\left(x_k\right)+\theta_{k+1}^\top\mathbb{E}\left[b\right]-\theta_k^T x_k,\\
&=\theta_{k+1}^T\mathbb{E}\left[b\right]+\sum_i\left(\mathbb{E}\left[W\right]^\top\theta_{k+1}\right)_i s\left(x_i\right)-\left(\theta_k\right)_i x_i.
\end{aligned} \tag{13}
$$

Maximizing the RHS subject to $l\leq x\leq u$, we obtain:

$$\theta_{k+1}^T\mathbb{E}\left[b\right]+\sum_i\max_{z\in[l_i,u_i]}\left(\mathbb{E}\left[W\right]^\top\theta_{k+1}\right)_i s\left(z\right)-\left(\theta_k\right)_i z.$$

where the maximization over z can be solved in closed form for most common activation functions s as shown in Dvijotham et al. [2018b].

So we can simply apply the deterministic algorithm to compute the closed-form solution of this problem.

D Relationship to Prior work

We establish connections between the functional Lagrangian framework and prior work on deterministic verification techniques based on Lagrangian relaxations and SDP relaxations.

D.1 Lagrangian Dual Approach

We assume that the network layers are deterministic layers of the form:

$$\begin{aligned} \forall\, k, k \text{ is odd } \ \pi_k(x) &= w_k x + b_k, \\ \forall\, k, k \text{ is even } \ \pi_k(x) &= s(x), \end{aligned} \tag{14}$$

where s is an element-wise activation function and that the specification can be written as:

$$\boldsymbol{\psi}(x_K) = c^T x_K. \tag{15}$$

Proposition 6 (Linear Multipliers)**.** *For a verification problem described by equations (14, 15), the functional Lagrangian framework with linear functional multipliers $\lambda_k(x) = \theta_k^T x$ is equivalent to the Lagrangian dual approach from Dvijotham et al. [2018b].*

Proof. The final layer problem is

$$\max_{x_K} c^T x_K - \theta_K^T x_K = \mathbf{1}^T \max\left((c-\theta_K)\odot l_K, (c-\theta_K)\odot u_K\right)$$

For even layers with $k < K$, the optimization problem is

$$\begin{aligned} &\max_{x\in[l_k,u_k]} \theta_{k+1}^T(w_k x + b_k) - \theta_k^T x = \theta_{k+1}^T b_k + \left(w_k^T\theta_{k+1} - \theta_k\right)^T x \\ &= \mathbf{1}^T \max\left(\left(w_k^T\theta_{k+1} - \theta_k\right)\odot l_k, \left(w_k^T\theta_{k+1} - \theta_k\right)\odot u_k\right) + \theta_{k+1}^T b_k \end{aligned}$$

For odd layers with $k < K$, the optimization problem is

$$\max_{x\in[l_k,u_k]} \theta_{k+1}^T s(x) - \theta_k^T x = \sum_i \max_{z\in[l_{ki},u_{ki}]} (\theta_{k+1})_i\, s(z) - (\theta_k)_i\, z$$

All these computations precisely match those from Dvijotham et al. [2018b], demonstrating the equivalence. □

D.2 SDP-cert

We assume that the network layers are deterministic layers of the form:

$$\forall\, k\ \pi_k(x) = \texttt{ReLU}(w_k x + b_k) \tag{16}$$

where s is an element-wise activation function and that the specification can be written as:

$$\boldsymbol{\psi}(x_K) = c^T x_K. \tag{17}$$

Proposition 7 (Quadratic Multipliers)**.** *For a verification problem described by equations (16, 17), the optimal value of the Functional Lagrangian Dual with*

$$\begin{aligned} \lambda_k(x) &= q_k^T x + \frac{1}{2} x^T Q_k x \quad k = 1, \ldots, K-1 \\ \lambda_K(x) &= q_K^T x \end{aligned}$$

and when an SDP relaxation is used to upper bound the inner maximization problems over g_k, is equal to the dual of the SDP relaxation from Raghunathan et al. [2018].

Proof. With quadratic multipliers of the form $\lambda_k(x) = q_k^T x + \frac{1}{2} x^T Q_k x \quad k = 1, \ldots, K-2$ and $\lambda_K(x) = q_K^T x$, the inner maximization problems for the intermediate layers are of the form:

$$\max_{\substack{x\in[l,u] \\ y=\texttt{ReLU}(wx+b)}} \tilde{q}^T y + \frac{1}{2} y^T \tilde{Q} y - q^T x - \frac{1}{2} x^T Q x,$$

where $l = l_k, u = u_k, \tilde{q} = q_{k+1}, \tilde{Q} = Q_{k+1}, q = q_k, Q = Q_k, w = w_k, b = b_k$. Let $x \in \mathbb{R}^n, y \in \mathbb{R}^m$ (n dimensional input, m dimensional output of the layer). Further, let $\tilde{l} = l_{k+1}, \tilde{u} = u_{k+1}$.

We can relax the above optimization problem to the following Semidefinite Program (SDP) (following Raghunathan et al. [2018]):

$$\max_{P} \tilde{q}^T P[y] + \frac{1}{2}\text{Trace}\left(\tilde{Q}P[yy^T]\right) - q^T P[x] - \frac{1}{2}\text{Trace}\left(QP[xx^T]\right) \tag{18a}$$

$$\text{Subject to } P = \begin{pmatrix} 1 & (P[y])^T & (P[x])^T \\ P[y] & P[yy^T] & P[xy^T] \\ P[x] & \left(P[xy^T]\right)^T & P[xx^T] \end{pmatrix} \in \mathbb{S}^{n+m+1}, \tag{18b}$$

$$P \succeq 0, \tag{18c}$$

$$\text{diag}\left(P[xx^T] - l\,(P[x])^T - P[x]u^T + lu^T\right) \leq 0, \tag{18d}$$

$$\text{diag}\left(P[yy^T] - \tilde{l}\,(P[y])^T - P[y]\tilde{u}^T + \tilde{l}\tilde{u}^T\right) \leq 0, \tag{18e}$$

$$P[y] \geq 0, P[w] \geq wP[x], \tag{18f}$$

$$\text{diag}\left(wP[xy^T]\right) + P[y] \odot b = \text{diag}\left(P[yy^T]\right). \tag{18g}$$

where the final constraint follows from the observation that $y \odot (y - wx - b) = 0$.

The above optimization problem resembles the formulation of Raghunathan et al. [2018] except that it only involves two adjacent layers rather than all the layers at once. Let Δ_k denote the feasible set given the constraints in the above optimization problem. Then, the formulation of Raghunathan et al. [2018] can be written as:

$$\max\ c^T y_K \tag{19a}$$

$$\text{subject to } P_k \in \Delta_k \quad k = 0, \ldots, K-1,\ l_K \leq y_K \leq u_K, \tag{19b}$$

$$P_{k+1}[xx^T] = P_k[yy^T] \quad k = 0, \ldots, K-2, \tag{19c}$$

$$P_{k+1}[x] = P_k[y] \quad k = 0, \ldots, K-2, \tag{19d}$$

$$y_K = P_{K-1}[y]. \tag{19e}$$

Note that in Raghunathan et al. [2018], a single large P matrix is used whose block-diagonal sub-blocks are P_k and the constraint $P \succeq 0$ is enforced. Due to the matrix completion theorem for SDPs [Grone et al., 1984, Vandenberghe and Andersen, 2015], it suffices to ensure postitive semidefiniteness of the sub-blocks rather than the full P matrix.

Dualizing the last three sets of constraints above with Lagrangian multipliers $\Theta_k \in \mathbb{S}^{n_k+n_{k+1}+1}, \theta_k \in \mathbb{R}^{n_k}$ and $\theta_K \in \mathbb{R}^{n_K}$, we obtain the following optimization problem:

$$\begin{aligned} \max\ & c^T y_K + \theta_K^T\left(-P_{K-1}[y] + y_K\right) + \sum_{k=0}^{K-2} \text{Trace}\left(\Theta_k\left(P_{k+1}[xx^T] - P_k[yy^T]\right)\right) \\ & + \sum_{k=0}^{K-2} \theta_k^T\left(P_{k+1}[x] - P_k[y]\right) \\ \text{subject to } & P_k \in \Delta_k \quad k = 0, \ldots, K-1, \\ & l_K \leq y_K \leq u_K. \end{aligned}$$

The objective decomposes over P_k and can be rewritten as:

$$\begin{aligned} \max_{l_K \leq y_K \leq u_K} & \left(c + \theta_K\right)^T y_K + \sum_{k=0}^{K-1} \max_{P_k \in \Delta_k} \text{Trace}\left(\Theta_{k-1}P_k[xx^T]\right) + \theta_{k-1}^T P_k[x] - \theta_k^T P_k[y] \\ & - \text{Trace}\left(\Theta_k P_k[yy^T]\right), \end{aligned} \tag{20}$$

with the convention that $\Theta_{K-1} = 0, \Theta_{-1} = 0, \theta_{-1} = 0$. If we set $Q_k = -\Theta_{k-1}, q_k = -\theta_{k-1}$ for $k = 1, \ldots, K$, then the optimization over P_k precisely matches the optimization in (18). Further, since λ_K is linear, the final layer optimization simply reduces to:

$$\max_{l_K \leq x_K \leq u_K} c^T x_K - q_K^T x_K,$$

which matches the first term in (20).

Thus, the functional Lagrangian framework with quadratic multipliers λ_k for $k = 1, \ldots, K-2$ and a linear multiplier for λ_K precisely matches the Lagrangian dual of (19) and since (19) is a convex optimization problem, strong duality guarantees that the optimal values must coincide.

□

E Additional Experimental Details

E.1 Robust OOD Detection on Stochastic Neural Networks

Inner Optimization. All inner problems have a closed-form as shown in section C.4, except for the last one, which is handled as follows.

The last inner problem can be formulated as:

$$\max_{x_K \in \mathcal{X}_K} \mu^\top \texttt{softmax}(x_K) + \nu^\top x_K, \tag{21}$$

where μ is a one-hot encoded vector and ν is a real-valued vector.

- Projected Gradient Ascent (Training):
  - Hyper-parameters: we use the Adam optimizer Kingma and Ba [2015], with a learning-rate of 1.0 and a maximum of 1000 iterations.
  - Stopping criterion: when all coordinates have either zero gradient, or are at a boundary with the gradient pointing outwards of the feasible set.
  - In order to help the gradient method find the global maximum, we use a heuristic for initialization, which consists of using the following two starting points for the maximization (and then to take the best of the corresponding two solutions found):
    1. Ignore affine part ($\nu = 0$), which gives a solution in closed form: set x_K at its upper bound at the coordinate where μ is 1, and at its lower bound elsewhere.
    2. Ignore softmax part ($\mu = 0$), which also gives a solution in closed form: set x_K at its upper bound at the coordinates where $\nu \geq 0$, and at its lower bound elsewhere.
- Evaluation: we use Algorithm 2 at evaluation time, which solves the maximization exactly.

Outer Optimization. We use the Adam optimizer, with a learning-rate that is initialized at 0.001 and divided by 10 every 250 steps. We run the optimization for a total of 1000 steps.

Gaussian-MLP. We use the ReLU MLP from [Wicker et al., 2020] that consists of 2 hidden layers of 128 units each. The models are available at `https://github.com/matthewwicker/ProbabilisticSafetyforBNNs`.

LeNet. We use the LeNet5 architecture with dropout applied to the last fully connected layer with a probability of 0.5. To make the bound-propagation simpler, we do not use max-pooling layers and instead increase the stride of convolutions.

VGG-X. For VGG-X (where X $\in \{2, 4, 8, 16, 32, 64\}$), the architecture can be described as:

- Conv 3x3, X filters, stride 1
- ReLU
- Conv 3x3, X filters, stride 2
- ReLU
- Conv 3x3, 2X filters, stride 2
- ReLU
- Conv 3x3, 2X filters, stride 2
- ReLU
- Flatten

- Linear with 128 output neurons
- Dropout with rate 0.2
- Linear with 10 output neurons

Hardware The verification of each sample is run on a CPU with 1-2 cores (and on each instance, BP and FL are timed on the same exact hardware configuration).

E.2 Adversarial Robustness for Stochastic Neural Networks

Inner Optimization. We use a similar approach as in Appendix E.1. For the final inner problem (corresponding to the objective which is a linear function of the softmax and the layer inputs), we run projected gradient ascent during the optimization phase and then use Algorithm 2 to solve the maximization exactly. For projected gradient ascent, because of the non-convexity of the problem, we use the following heuristics to try and find the global maximum:

- Black-box attack (1st phase): we use the `Square` adversarial attack Andriushchenko et al. [2020], with 600 iterations, 300 random restarts and learning-rate of 0.1.
- Fine-tuning (2nd phase): We then choose the best attack from the restarts, and employ projected gradient ascent, with a learning-rate of 0.1 and 100 iterations to fine-tune further.

Model Parameters. We use the 1 and 2 layer ReLU MLPs from [Wicker et al., 2020]. The models are available at `https://github.com/matthewwicker/ProbabilisticSafetyforBNNs`.

Outer Optimization. We use the Adam optimizer, with a learning-rate that is initialized at 0.001 and divided by 10 every 1000 steps. We run the optimization for a total of 3000 steps.

Hardware All experiments were run on a P100 GPU.

E.3 Distributionally Robust OOD Detection

Model. We train networks on MNIST using the code from `https://gitlab.com/Bitterwolf/GOOD` with the CEDA method, and with the default hyperparameters. We train a CNN with `ReLU` activations the following layers:

- Conv 4x4, 16 filters, stride 2, padding 2 on both sides
- ReLU
- Conv 4x4, 32 filters, stride 1, padding 1 on both sides
- Relu
- Flatten
- Linear with 100 output neurons
- Relu
- Linear with 10 output neurons

Outer Optimization. For the outer loop of the verification procedure, we use Adam for 100k steps. The learning-rate is initially set to 0.0001 and then divided by 10 after 60k and 80k steps.

Hardware We run the experiments for this section on a CPU with 2-4 cores.

F Additional Results with Interval Bound Propagation for Bilinear Operations

F.1 Robust OOD Detection for Stochastic Neural Networks

We repeat the experiments in Section 5.1 where we use IBP to handle bound-propagation through the layers where bilinear propagation is required (because of bounds coming from both the layer

Table 4: Robust OOD Detection: MNIST vs EMNIST (MLP and LeNet) and CIFAR-10 vs CIFAR-100 (VGG-*). BP: Bound-Propagation (baseline), using IBP instead of Bunel et al. [2020] for bilinear operations; FL: Functional Lagrangian (ours). The reported times correspond to the median of the 500 samples.

OOD Task	Model	#neurons	#params	ϵ	Time (s)		GAUC (%)		AAUC (%)
					BP	FL	BP	FL	
(E)MNIST	MLP	256	2k	0.01	1.1	+13.1	55.4	**67.5**	86.9
				0.03	1.2	+13.4	38.7	**54.5**	88.6
				0.05	1.3	+17.7	19.1	**36.0**	88.8
(E)MNIST	LeNet	0.3M	0.1M	0.01	50.1	+13.1	0.0	**28.4**	71.6
				0.03	54.7	+13.7	0.0	**11.7**	57.6
				0.05	79.4	+24.8	0.0	**2.3**	44.0
CIFAR	VGG-16	3.0M	83k	0.001	426.4	+21.4	0.0	**21.7**	60.9
	VGG-32	5.9M	0.2M	0.001	1035.2	+21.3	0.0	**23.8**	64.7
	VGG-64	11.8M	0.5M	0.001	8549.7	+42.1	0.0	**28.6**	67.4

Table 5: Adversarial Robustness for different BNN architectures trained on MNIST from Wicker et al. [2020]. BP: Bound-Propagation (baseline), using IBP instead of LBP for bilinear operations; FL: Functional Lagrangian (ours). The accuracy reported for FL and BP is the % of samples we can certify as robust with probability 1. For each model, we report results for the first 500 test-set samples.

#layers	ϵ	#neurons	BP Acc. (%)	FL Acc. (%)	BP Time (s)	FL Time (s)	Adv Acc (%)
1	0.025	128	43.8	**65.2**	1.3	+353.3	82.6
		256	40.6	**64.6**	1.4	+431.3	82.6
		512	35.0	**57.0**	1.3	+357.1	82.8
2	0.001	256	29.4	**36.9**	1.6	+439.6	79.4
		512	46.0	**63.4**	1.7	+433.8	89.2
		1024	18.4	**19.6**	1.6	+440.9	74.8

inputs and the layer parameters due to the stochasticity of the model) instead of Bunel et al. [2020]. Bunel et al. [2020] usually results in significantly tighter bounds compared to IBP but we note that for MNIST-CNN and CIFAR-CNN, we expect IBP to perform competitively as the bilinear bound propagation is only applied for a single layer (dropout). The results are presented in Table 4, and we find that even while using IBP as the bound-propagation method, our framework provides significantly stronger guarantees.

F.2 Adversarial Robustness for Stochastic Neural Networks

For the verification tasks considered in Section 5.2, we use IBP instead of the tighter LBP as the bound-propagation method and report results in Table 2. We find that, similar to Section 5.2, our framework is able to significantly improve on the guarantees the bound-propagation baseline is able to provide.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] Our theoretical framework is explained in Section 3 and our empirical results are detailed in Section 5.
   (b) Did you describe the limitations of your work? [Yes] See Section 6.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Section 6.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] Formal statements are available in Section 3 and Appendix C.
   (b) Did you include complete proofs of all theoretical results? [Yes] Formal proofs are available in Section B and Appendix C.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] We have included the URL at which the code will be released.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] We provide experimental details in Appendix E.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] We follow standard practice in the neural network verification community, and our evaluation is performed on hundreds of samples to ensure that the results are statistically significant.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] We have included all runtimes and hardware platforms used.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]