Thanks for the comments! Note the reviewers all found Bayesian Layers (BL) to have significant practical impact. R2 and R3 also found the design to be novel, while R4's major criticism is that the design is too similar to Aboleth's. (We hope to convince R4 that this is not true below). Below we answer major comments; we'll fix the minor ones.

REVIEWER 2 **Pyro's random module** Pyro can't be extended with recent estimators. It can only change how weights are instantiated in deterministic layers. To change the computation itself requires a design similar to BL's.

"When it comes to practicality, I find that BL is sufficiently different, more consistent and more extensive...Having said that, comparison to Pyro would be useful." Thanks! In our revision, we'll include a Pyro implementation of Figure 1 (the Bayesian RNN). Note Appendix A includes implementations in raw TF and Edward.

extended snippet separating model and inference We'll link to an end-to-end script and which involves real data.

"I suggest a background section listing the basic topics before diving deeper into the details." That's a good idea! We'll clean up the presentation and start with basic topics as background.

larger discussion about limitations of Bayesian models to DNNs Great idea. We'll add to the discussion. BL is indeed a reply to the software limitation. More open limitations exist: targeted applications; practical tricks; baselines.

REVIEWER 3 **"I am slightly puzzled by the insistence on referring to the module as catering for 'neural networks'."** We can revise the title to "functional uncertainty" or "uncertainty-aware functions." Let us know!

MXFusion We're happy to mention MXFusion. Studying the paper and code, MXFusion's "probabilistic module" is highly related in tying models with an inference algorithm. Unfortunately, I'm not clear on how their inference modularity composes. For example, their support for deep GPs is preliminary. Their GitHub branch (**url**) suggests that this involves a custom class rather than BL's approach of simply composing variational GP layers.

Cutajar et al (2017) It's indeed relevant! We'll cite it.

clearer summary of contributions We'll add a Contributions section. In summary, BL is the first to: contribute a unifying design across uncertainty-aware functions (BNNs, GPs, stochastic outputs, reversible layers); enable a BNN/GP design as part of existing semantics+ecosystems; and demonstrate practical examples on complex environments.

more benchmarks. Our current experiments apply BNNs with features not present in previous libraries. We decided not to benchmark against Pyro and MXFusion because this would reduce to a benchmark of their backends. For the Bayesian Transformer, we have benchmarked Examples/Sec under libraries with the same backend however (i.e., vanilla TF and Edward). In both graph and eager mode, runtimes are the same with negligible differences from simulation variance; we'll include more details in the paper. See also the comment to R4 about additional comparisons.

"I am still curious to understand whether say, the DGP of Salimbeni et al. (2017) which can be constructed here, is just as good as the original implementation." It's mathematically the same as Salimbeni et al. (2017). Note this is not the same, however, as Damianou and Lawrence (2013). The latter may be what MXFusion is implementing by constructing a separate (not composable) class for DGPs; BL's approach simply composes variational GP layers.

REVIEWER 4 **Aboleth.** Aboleth uses a different design, ending up with a less flexible framework that makes it more challenging to use for research. We'll add the following points to the revision:

- **For BNNs, Aboleth hardcodes Gaussian priors and Gaussian posteriors.** BL supports arbitrary priors and posteriors, including implicit distributions and hypernetworks. BL also supports arbitrary estimators, including local reparameterization and deterministic VI (Sec 2.1), and probabilistic programming for dynamic models and inference (Sec 2.5). It's not clear how to extend Aboleth's design to this broader support without incorporating BL's design.
- **For GPs, Aboleth only supports random feature approximations.** It does not support exact GPs or variational GPs. For stochastic output layers and reversible layers, Aboleth does not support them. This makes Aboleth primarily a BNN library. In contrast, BL implements a unifying design across uncertainty-aware functions.
- **Aboleth creates a new neural network language.** BL instead considers how to augment the existing Keras semantics, unifying deterministic and stochastic functions. (In contrast, for example, `aboleth.DenseVariational` has a separate design from `aboleth.Dense`.) BL's benefit is building on an existing ecosystem, leveraging the efficiently optimized deterministic layer computations as well as compatibility across TensorFlow libraries.

not following up on more recent estimators. Flipout and Deterministic VI are already available in BL. Implementation-wise, they subclass, e.g., DenseReparameterization and only reimplement the `call()` method (described in Sec 2.1).

toolkit is inflexible: "I am under the impression that only VI with Gaussian prior and variational posterior (with diagonal covariance) is supported." Footnote 2 describes non-Gaussian priors and posteriors. The only constraint is that the layer initializer returns a (stochastic) Tensor representing a sample; the layer regularizer can utilize prior/posterior assumptions as necessary. For example, for an implicit prior and posterior, have the initializer simulate from q; have the regularizer compute a density ratio loss involving only the sample and a trainable discriminator.

"comparison with deterministic in term of training time and memory overhead." Sec 3.1's deterministic Transformer takes 13 hours; the Bayesian Transformer takes 16 hours and 2 extra gb per TPU. Sec 3.2's deterministic dynamics model takes 20 hours, 8 gb; the Bayesian dynamics model takes 22 hours; 8 gb. We'll add to the revision.