Uncertainty Baselines: Benchmarks for Uncertainty & Robustness in Deep Learning

Zachary Nado znado@google.com*
Neil Band neil.band@cs.ox.ac.uk†
Mark Collier markcollier@google.com*
Josip Djolonga josipd@google.com*
Michael W. Dusenberry dusenberrymw@google.com*
Sebastian Farquhar sebastian.farquhar@cs.ox.ac.uk†
Qixuan Feng qixuan.feng1@gmail.com†
Angelos Filos angelos.filos@cs.ox.ac.uk†
Marton Havasi mh740@cam.ac.uk‡
Rodolphe Jenatton rjenatton@google.com*
Ghassen Jerfel ghassen@google.com*
Jeremiah Liu jereliu@google.com§
Zelda Mariet zmariet@google.com*
Jeremy Nixon jnixon2@gmail.com*
Shreyas Padhy shreyaspadhy@google.com*
Jie Ren jjren@google.com*
Tim G. J. Rudner tim.rudner@cs.ox.ac.uk†
Yeming Wen ywen@utexas.edu¶
Florian Wenzel florianwenzel@google.com*
Kevin Murphy kpmurphy@google.com*
D. Sculley dsculley@google.com*
Balaji Lakshminarayanan balajiln@google.com*
Jasper Snoek jsnoek@google.com*
Yarin Gal yarin@cs.ox.ac.uk†
Dustin Tran trandustin@google.com*

Abstract

High-quality estimates of uncertainty and robustness are crucial for numerous real-world applications, especially for deep learning which underlies many deployed ML systems. The ability to compare techniques for improving these estimates is therefore very important for research and practice alike. Yet, competitive comparisons of methods are often lacking due to a range of reasons, including: compute availability for extensive tuning, incorporation of sufficiently many baselines, and concrete documentation for reproducibility. In this paper we introduce Uncertainty Baselines: high-quality implementations of standard and state-of-the-art deep learning methods on a variety of tasks. As of this writing, the collection spans 19 methods across 9 tasks, each with at least 5 metrics. Each baseline is a self-contained experiment pipeline with easily reusable and extendable components. Our goal is to provide immediate starting points for experimentation with new methods or applications. Additionally we provide model checkpoints, experiment outputs as Python notebooks, and leaderboards for comparing results. https://github.com/google/uncertainty-baselines

*Google Research, Brain Team †OATML, Department of Computer Science, University of Oxford ‡Department of Engineering University of Cambridge §Department of Biostatistics, Harvard University ¶University of Texas at Austin

1. Introduction

Baselines on standardized benchmarks are crucial to machine learning research for measuring whether new ideas yield meaningful progress. However, reproducing the results from previous works can be extremely challenging, especially when only reading the paper text (Sinha et al., 2020; D'Amour et al., 2020). Having access to the code for experiments is more useful, assuming it is well-documented and maintained. But even this is not enough. In fact, in retrospective analyses over a collection of works, authors often find that a simpler baseline works best in practice, due to flawed experiment protocols or insufficient tuning (Melis et al., 2017; Kurach et al., 2019; Bello et al., 2021; Nado et al., 2021).

There is a wide spectrum of experiment artifacts made available in papers. A popular approach is a GitHub dump of code used to run experiments, albeit lacking documentation and tests. At best, papers might provide actively maintained repositories with examples, model checkpoints, and ample documentation to extend the work. A single paper can only go so far however: without community standards, each paper's codebase differs in experimental protocol and code organization, making it difficult to compare across papers within a common benchmark, let alone build jointly on top of multiple papers.

To address these challenges, we created the Uncertainty Baselines library. It provides high-quality implementations of baselines across many uncertainty and out-of-distribution robustness tasks. Each baseline is designed to be self-contained (i.e., minimal dependencies) and easily extensible. We provide numerous artifacts in addition to the raw code so that others can adapt any baseline to suit their workflow.

Related work. OpenAI Baselines (Dhariwal et al., 2017) is work in similar spirit for reinforcement learning. Prior work on uncertainty and robustness benchmarks include Riquelme et al. (2018); Filos et al. (2019); Hendrycks and Dietterich (2019); Ovadia et al. (2019); Dusenberry et al. (2020b). These all introduce a new task and evaluate a variety of baselines on that task. In practice, they are unmaintained, focusing on experimental insights rather than the codebase as the contribution. Our work provides an extensive set of benchmarks (in several cases, unifying the above ones), has a larger set of baselines across these benchmarks, and focuses on designing scalable, forkable, and well-tested code.

2. Uncertainty Baselines

Uncertainty Baselines sets up each benchmark as a choice of base model, training dataset, and a suite of evaluation metrics.

1. Base models (architectures) include Wide ResNet 28-10 (Zagoruyko and Komodakis, 2016), ResNet-50 (He et al., 2016), BERT (Devlin et al., 2018), and simple MLPs.
2. Training datasets include standard machine learning datasets – CIFAR (Krizhevsky et al., a,b), ImageNet (Russakovsky et al., 2015), and UCI (Dua and Graff, 2017) – as well as more real-world problems – Clinc Intent Detection (Larson et al., 2019), Kaggle's Diabetic Retinopathy Detection (Filos et al., 2019), and Wikipedia Toxicity (Wulczyn et al., 2017). These span modalities such as tabular, text, and images.
3. Evaluation includes predictive metrics such as accuracy, uncertainty metrics such as selective prediction and calibration error, compute metrics such as inference latency, and performance under in- and out-of-distribution datasets.

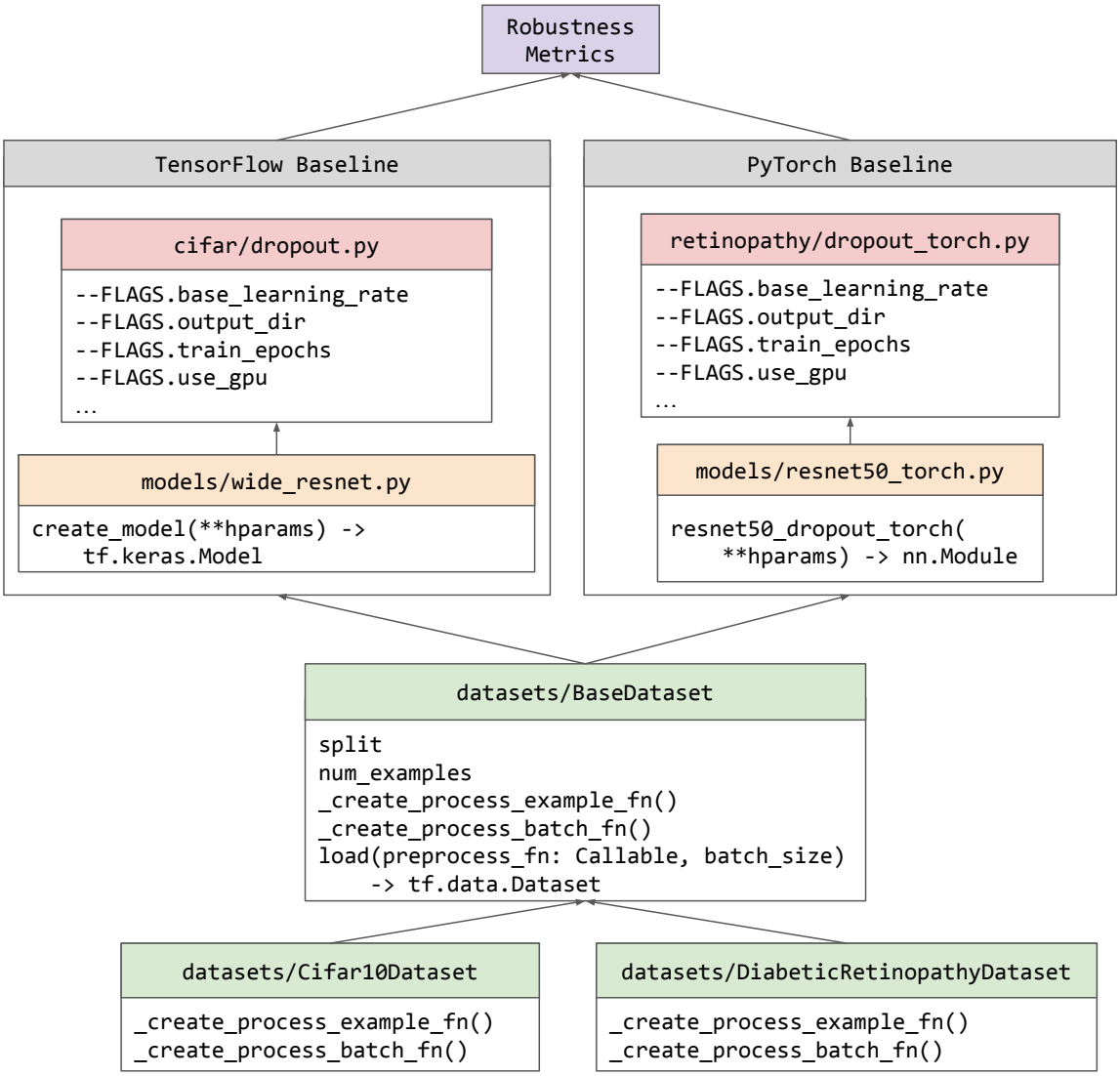

Figure 1: The structure for an experiment under the TensorFlow or Pytorch backend. One instantiates a dataset (`Cifar10Dataset` or `DiabeticRetinopathyDataset`) and model (`wide_resnet` or `resnet50_torch`) within an end-to-end training script. After training, one inputs saved model checkpoints into Robustness Metrics for evaluation.

As of this writing, we provide a total of 83 baselines, comprising 19 methods sweeping over standard and more recent strategies over 9 benchmarks.

Modularity. In order to optimize for researchers to easily experiment on baselines (specifically, fork them), we designed the baselines to be as modular as possible and with minimal non-standard dependencies. API-wise, Uncertainty Baselines provides little to no abstractions: datasets are light wrappers around TensorFlow Datasets (TFDS Team), models are Keras models, and training/test logic is in raw TensorFlow (Abadi et al., 2015)

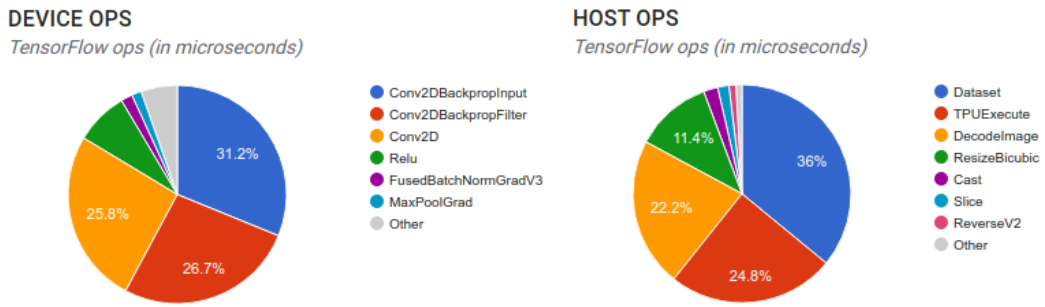

Figure 2: Performance analysis of a MIMO baseline on a TPUv3-32 using the TensorFlow Profiler. The runtime is optimized, bound only by model operations, an irreducible bottleneck for a given baseline. Our implementations have 100% utilization of the TPU devices.

This allows new users to more easily run individual examples, or incorporate our datasets and/or models into their libraries. For out-of-distribution evaluation, we plug our trained models into Robustness Metrics (Djolonga et al., 2020). Figure 1 illustrates how the modules fit together.

Framework. Uncertainty Baselines is framework-agnostic. The dataset and metric modules are NumPy-compatible, and interoperate in a performant manner with modern deep learning frameworks including TensorFlow, Jax, and PyTorch. For example, our baselines on the JFT-300M dataset use raw JAX, and we include a PyTorch Monte Carlo Dropout baseline on the Diabetic Retinopathy dataset. In practice, for ease of code and performance comparison, we choose a specific backend for each benchmark and develop all baselines under that backend (most often TensorFlow). Our Jax and PyTorch baselines demonstrate that implementation with other frameworks is supported and straightforward.

Hardware. All baselines run on CPU, GPU, or Google Cloud TPUs. Baselines are optimized for a default hardware configuration and often assume a memory requirement and number of chips (e.g., 1 GPU, or TPUv2-32) in order to reproduce the results. We employ the latest coding practices to fully utilize accelerator chips (Figure 2) so researchers can leverage the most performant baselines.

Hyperparameters. Hyperparameters and other experiment configuration values easily number in the dozens for a given baseline. Uncertainty Baselines uses standard Python flags to specify hyperparameters, setting default values to reproduce best performance. Flags are simple, require no additional framework, and are easy to plug into other pipelines or extend. We also document the protocol to properly tune and evaluate baselines—a common source of discrepancy in papers.

Reproducibility. All modules include testing, and all results are reported over multiple seeds. Computing metrics on trained models can be prohibitively expensive let alone training from scratch. Therefore we also provide TensorBoard dashboards which include all training, tuning, and evaluation metrics. An example can be found here.

3. Results

To provide an example of Uncertainty Baselines' features, we display baselines available on 1 of 9 tasks: ImageNet. Figure 3 displays accuracy and calibration error across 8 base-

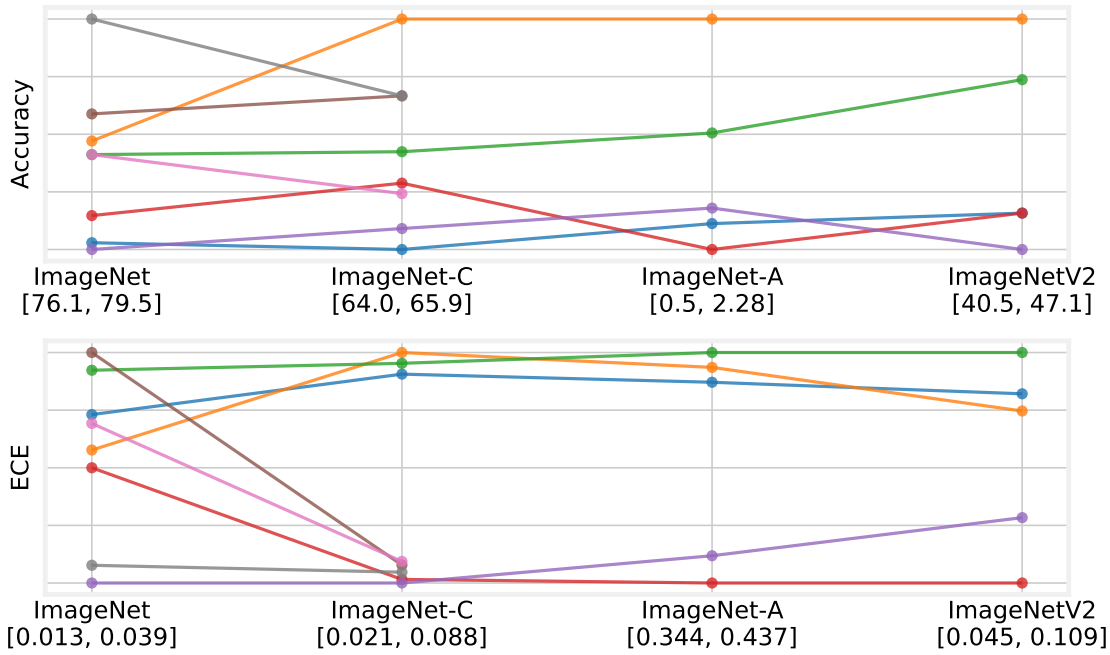

Figure 3: 8 baselines evaluated on ImageNet, ImageNet-C, ImageNet-A, and ImageNetV2 (matched frequency variant). **(top)** Top-1 accuracy. **(bottom)** Expected calibration error. Results demonstrate the many baselines available with competitive performance.

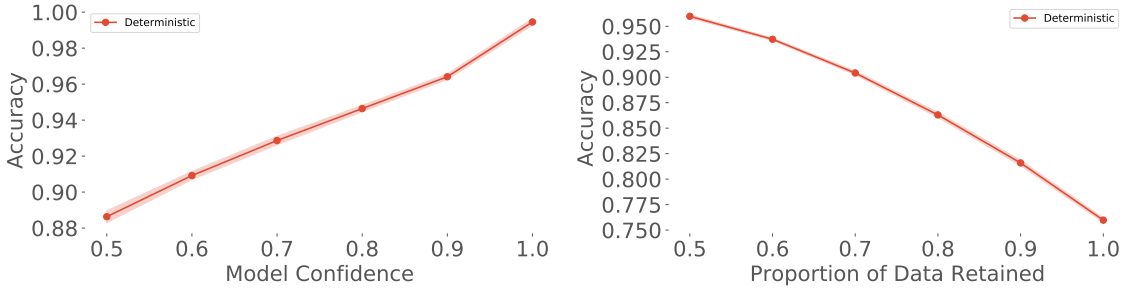

Figure 4: ImageNet baseline applied to deferred prediction. In this task, one defers predictions according to the model's confidence (**left**) or a desired data retention rate (**right**).

lines, evaluated on in- and out-of-distribution.[1] Figure 4 provides an example of applying such baselines to a downstream task. Overall, the results demonstrate only a sampling of the repository's capabilities. We are excited to see new research already building on the baselines.

[1] We omit a legend to avoid drawing comparisons among which specific baselines perform best. See our full leaderboards to draw those insights at `baselines/imagenet/README.md`.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org.

Irwan Bello, William Fedus, Xianzhi Du, Ekin D Cubuk, Aravind Srinivas, Tsung-Yi Lin, Jonathon Shlens, and Barret Zoph. Revisiting resnets: Improved training and scaling strategies. *arXiv preprint arXiv:2103.07579*, 2021.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In *International Conference on Machine Learning*, pages 1613–1622. PMLR, 2015.

Luigi Carratino, Moustapha Cissé, Rodolphe Jenatton, and Jean-Philippe Vert. On mixup regularization. *arXiv preprint arXiv:2006.06049*, 2020.

Mark Collier, Basil Mustafa, Efi Kokiopoulou, Rodolphe Jenatton, and Jesse Berent. Correlated input-dependent label noise in large-scale image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2021.

Jigsaw Conversation AI. Toxic comment classification challenge. https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge, 2017.

Alexander D'Amour, Katherine Heller, Dan Moldovan, Ben Adlam, Babak Alipanahi, Alex Beutel, Christina Chen, Jonathan Deaton, Jacob Eisenstein, Matthew D Hoffman, et al. Underspecification presents challenges for credibility in modern machine learning. *arXiv preprint arXiv:2011.03395*, 2020.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, Yuhuai Wu, and Peter Zhokhov. Openai baselines. https://github.com/openai/baselines, 2017.

Josip Djolonga, Frances Hubis, Matthias Minderer, Zack Nado, Jeremy Nixon, Rob Romijnders, Dustin Tran, and Mario Lucic. Robustness Metrics, 2020. URL https://github.com/google-research/robustness_metrics.

Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

Michael Dusenberry, Ghassen Jerfel, Yeming Wen, Yian Ma, Jasper Snoek, Katherine Heller, Balaji Lakshminarayanan, and Dustin Tran. Efficient and scalable bayesian neural nets with rank-1 factors. In *International conference on machine learning*, pages 2782–2792. PMLR, 2020a.

Michael W Dusenberry, Dustin Tran, Edward Choi, Jonas Kemp, Jeremy Nixon, Ghassen Jerfel, Katherine Heller, and Andrew M Dai. Analyzing the role of model uncertainty for electronic health records. In *Proceedings of the ACM Conference on Health, Inference, and Learning*, pages 204–213, 2020b.

Sebastian Farquhar, Michael A Osborne, and Yarin Gal. Radial bayesian neural networks: Beyond discrete support in large-scale bayesian deep learning. In *International Conference on Artificial Intelligence and Statistics*, pages 1352–1362. PMLR, 2020.

Angelos Filos, Sebastian Farquhar, Aidan N Gomez, Tim GJ Rudner, Zachary Kenton, Lewis Smith, Milad Alizadeh, Arnoud de Kroon, and Yarin Gal. A systematic comparison of bayesian deep learning robustness in diabetic retinopathy tasks. *arXiv preprint arXiv:1912.10481*, 2019.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR, 2016.

Marton Havasi, Rodolphe Jenatton, Stanislav Fort, Jeremiah Zhe Liu, Jasper Snoek, Balaji Lakshminarayanan, Andrew M Dai, and Dustin Tran. Training independent subnetworks for robust prediction. *arXiv preprint arXiv:2010.06610*, 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261*, 2019.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research). a. URL http://www.cs.toronto.edu/~kriz/cifar.html.

Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-100 (canadian institute for advanced research). b. URL http://www.cs.toronto.edu/~kriz/cifar.html.

Karol Kurach, Mario Lučić, Xiaohua Zhai, Marcin Michalski, and Sylvain Gelly. A large-scale study on regularization and normalization in gans. In *International Conference on Machine Learning*, pages 3581–3590. PMLR, 2019.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *arXiv preprint arXiv:1612.01474*, 2016.

Stefan Larson, Anish Mahendran, Joseph J Peper, Christopher Clarke, Andrew Lee, Parker Hill, Jonathan K Kummerfeld, Kevin Leach, Michael A Laurenzano, Lingjia Tang, et al. An evaluation dataset for intent classification and out-of-scope prediction. *arXiv preprint arXiv:1909.02027*, 2019.

Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010. URL http://yann.lecun.com/exdb/mnist/.

Jeremiah Zhe Liu, Zi Lin, Shreyas Padhy, Dustin Tran, Tania Bedrax-Weiss, and Balaji Lakshminarayanan. Simple and principled uncertainty estimation with deterministic deep learning via distance awareness. *arXiv preprint arXiv:2006.10108*, 2020.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Gábor Melis, Chris Dyer, and Phil Blunsom. On the state of the art of evaluation in neural language models. *arXiv preprint arXiv:1707.05589*, 2017.

Jishnu Mukhoti, Viveka Kulharia, Amartya Sanyal, Stuart Golodetz, Philip Torr, and Puneet Dokania. The intriguing effects of focal loss on the calibration of deep neural networks. 2019.

Zachary Nado, Justin M Gilmer, Christopher J Shallue, Rohan Anil, and George E Dahl. A large batch optimizer reality check: Traditional, generic optimizers suffice across batch sizes. *arXiv preprint arXiv:2102.06356*, 2021.

Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, David Sculley, Sebastian Nowozin, Joshua V Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. *arXiv preprint arXiv:1906.02530*, 2019.

Carlos Riquelme, George Tucker, and Jasper Snoek. Deep bayesian bandits showdown: An empirical comparison of bayesian deep networks for thompson sampling. *arXiv preprint arXiv:1802.09127*, 2018.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

Koustuv Sinha, Joelle Pineau, Jessica Forde, Rosemary Nan Ke, and Hugo Larochelle. Neurips 2019 reproducibility challenge. *ReScience C*, 6(2):11, 2020.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147. PMLR, 2013.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper

with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pages 6105–6114. PMLR, 2019.

TFDS Team. TensorFlow Datasets, a collection of ready-to-use datasets. https://www.tensorflow.org/datasets.

Yeming Wen, Dustin Tran, and Jimmy Ba. Batchensemble: an alternative approach to efficient ensemble and lifelong learning. *arXiv preprint arXiv:2002.06715*, 2020.

Florian Wenzel, Jasper Snoek, Dustin Tran, and Rodolphe Jenatton. Hyperparameter ensembles for robustness and uncertainty quantification. *arXiv preprint arXiv:2006.13570*, 2020.

Ellery Wulczyn, Nithum Thain, and Lucas Dixon. Ex machina: Personal attacks seen at scale. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, pages 1391–1399, Republic and Canton of Geneva, CHE, 2017. International World Wide Web Conferences Steering Committee. ISBN 9781450349130. doi: 10.1145/3038912.3052591. URL https://doi.org/10.1145/3038912.3052591.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

Appendix A. Supported Baselines

Appendix B. Dataset Details

For CIFAR10 and CIFAR100, we padded the images with 4 pixels of 0's before doing a random crop to 32x32 pixels, followed by a left-right flip with 50% chance. For ImageNet, we used ResNet preprocessing as described in He et al. (2016), but also support the common Inception preprocessing from Szegedy et al. (2015). All preprocessing is deterministic given a random seed, using `tf.random.experimental.stateless_split` and `tf.random.experimental.stateless_fold_in`. For the Diabetic Retinopathy benchmarks we used the Kaggle competition dataset as in Filos et al. (2019).

Appendix C. Model Details

For CIFAR10 and CIFAR100 we provide methods based on the Wide ResNet models, typically the Wide ResNet-28 size (Zagoruyko and Komodakis, 2016). For ImageNet and the Diabetic Retinopathy benchmarks, we provide methods based on the ResNet-50 model (He et al., 2016). For ImageNet we additionally use methods based on the EfficientNet models (Tan and Le, 2019). For the Toxic Comments and CLINC Intent Detection benchmarks, our methods are based on the BERT-Base model (Devlin et al., 2018).

Appendix D. Hyperparameter Tuning

All image benchmarks were trained with Nesterov momentum (Sutskever et al., 2013), except for the EfficientNet models which use RMSProp with $\rho = 0.9, \epsilon = 10^{-3}$. The text benchmarks were trained with the AdamW optimizer (Loshchilov and Hutter, 2017) with a $\beta_2 = 0.999, \epsilon = 10^{-6}$. Unless otherwise noted, the image benchmarks used a linear warmup followed by a stepwise decay schedule, except for the EfficientNet models which used a linear warmup followed by an exponential decay. The text benchmarks used a linear warmup followed by a linear decay.

For the CIFAR10, CIFAR100, ImageNet, Toxic Comments, and CLINC Intent Detection benchmarks, the papers for each method contain their tuning details.

Diabetic Retinopathy benchmark tuning details. For the Diabetic Retinopathy benchmark, we also provide our tuning results so that others can more easily retune their own methods. We conducted two rounds of quasirandom search on several hyperparameters (learning rate, momentum, dropout, variational posteriors, L2 regularization), where the first round was a heuristically-picked larger search space and the second round was a hand-tuned smaller range around the better performing values. Each round was for 50 trials, and the final hyperparameters were selected using the final validation AUC from the second tuning round. We finally retrained this best hyperparameter setting on the combined train and validation sets.

Dataset	Method
CIFAR (Krizhevsky, 2009)	BatchEnsemble (Wen et al., 2020)
	Hyper-BatchEnsemble (Wenzel et al., 2020)
	MIMO (Havasi et al., 2020)
	Rank-1 BNN (Gaussian) (Dusenberry et al., 2020a)
	Rank-1 BNN (Cauchy)
	SNGP (Liu et al., 2020)
	MC-Dropout (Gal and Ghahramani, 2016)
	Ensemble (Lakshminarayanan et al., 2016)
	Hyper-deep ensemble (Wenzel et al., 2020)
	Variational Inference (Blundell et al., 2015)
	Heteroscedastic (Collier et al., 2021)
CLINC (Larson et al., 2019)	SNGP
	MC-Dropout
	Ensemble
Diabetic Retinopathy Detection (Filos et al., 2019)	MC-Dropout
	Ensemble
	Radial Bayesian Neural Networks (Farquhar et al., 2020)
	Variational Inference
ImageNet (Russakovsky et al., 2015)	MixUp (Carratino et al., 2020)
	BatchEnsemble
	Hyper-BatchEnsemble
	MIMO
	Rank-1 BNN (Gaussian)
	Rank-1 BNN (Cauchy)
	SNGP
	MC-Dropout
	Ensemble
	Hyper-deep ensemble
	Variational Inference
	Heteroscedastic
MNIST (LeCun and Cortes, 2010)	Variational Inference
Toxic Comments Detection (Conversation AI, 2017)	SNGP
	MC-Dropout
	Ensemble
	Focal Loss (Mukhoti et al., 2019)
UCI (Dua and Graff, 2017)	Variational Inference

Table 1: Currently implemented methods for each dataset, in addition to a deterministic baseline. See repository for a more updated list.

Appendix E. Open-Source Data

The tuning and final metrics data for the Diabetic Retinopathy benchmarks can be found at the following URLs:

- Deterministic First Tuning
- Deterministic Final Tuning
- Deterministic 10 seeds
- Dropout First Tuning
- Dropout Final Tuning
- Dropout 10 seeds
- Variational Inference First Tuning
- Variational Inference Final Tuning
- Variational Inference 10 seeds
- Radial BNN First Tuning
- Radial BNN Final Tuning
- Radial BNN 10 seeds