Bayesian Neural Networks for Cellular Image Classification and Uncertainty Analysis

PREPRINT

Giacomo Deodato[*]
Eurecom - Graduate School and Research Center
Novartis Institutes for Biomedical Research
deodato@eurecom.fr

Christopher Ball[*]
Novartis Institutes for Biomedical Research
christopher.ball@novartis.com

Xian Zhang[*]
Novartis Institutes for Biomedical Research
xian-1.zhang@novartis.com

Abstract

Over the last decades, deep learning models have rapidly gained popularity for their ability to achieve state-of-the-art performances in different inference settings. Deep neural networks have been applied to an increasing number of problems spanning different domains of application. Novel applications define a new set of requirements that transcend accurate predictions and depend on uncertainty measures. The aims of this study are to implement Bayesian neural networks and use the corresponding uncertainty estimates to perform predictions and dataset analysis. We identify two main advantages in modeling the predictive uncertainty of deep neural networks performing classification tasks. The first is the possibility to discard highly uncertain predictions to be able to guarantee a higher accuracy of the remaining predictions. The second is the identification of unfamiliar patterns in the data that correspond to outliers in the model representation of the training data distribution. Such outliers can be further characterized as either corrupted observations or data belonging to different domains. Both advantages are well demonstrated with the benchmark datasets. Furthermore we apply the Bayesian approach to a biomedical imaging dataset where cancer cells are treated with diverse drugs, and show how one can increase classification accuracy and identify noise in the ground truth labels with uncertainty analysis.

1 Introduction

Deep neural networks have seen a dramatic increase in popularity in recent years, due to their outstanding performances in complex prediction tasks [Krizhevsky et al., 2012, LeCun et al., 2015]. The main drawback of neural networks lies in their lack of interpretability (they are often deemed as "black boxes" [Benítez et al., 1997, Duch, 2003, Shrikumar et al., 2017, Lundberg and Lee, 2017]) and their dependence on point estimates of their parameters. Despite their ability to outperform simpler models, a single prediction score (i.e. the accuracy of the prediction) is not sufficient for a variety of tasks and domain applications [Ghahramani, 2015]. Modeling applications such as healthcare require an additional feature to the prediction score, that is a measure of confidence that reflects the uncertainty of the predictions. For example, a neural network performing diagnosis of brain tumors by analyzing magnetic resonance images needs a way to express the ambiguity of an image in the same way as a doctor may express uncertainty and ask for experts help. Moreover, predictive uncertainty provides further insights about the data because more certain predictions correspond to cleaner data both from a technical and a contextual point of view.

[*]to whom correspondence should be addressed.

The architectures used to perform classification tasks have a number of output nodes equal to the number of possible classes. The class with the higher output value corresponds to the predicted label. These output values are usually transformed using the softmax activation function so that they lay in the range [0, 1] and their sum equals one. Because of these properties the output of the neural network is often treated as a probability distribution. However, despite there is a correlation between the accuracy of the prediction and this confidence score, that is the probability value of the output, this should not lead to think that this output feature is an appropriate measure of uncertainty as this would show that the model makes overconfident predictions most of time [Gal and Ghahramani, 2016].

Instead of the described prediction score, we analyzed data by means of the *predictive* uncertainty, that can be decomposed into *epistemic* uncertainty, which stems from model's parameters as well as the specific architecture of the model, and *aleatoric* uncertainty, which depends on the noise of the observations [Der Kiureghian and Ditlevsen, 2009].

Epistemic uncertainty is useful when big data sets are not available because these are the scenarios where the model's parameters are more uncertain. Furthermore, it is key to understand *out-of-distribution* samples, observations that are different from the training data, e.g. an image of a letter when the training data are images of digits, as well as to find novel features in the input data. While epistemic uncertainty can be heavily reduced by feeding the model a large enough amount of data, aleatoric uncertainty cannot be explained away [Kendall and Gal, 2017]. For this reason, aleatoric uncertainty is useful in big data scenarios where the epistemic component of uncertainty disappears, potentially leading the model to overconfident predictions. Furthermore, it provides a measure of the diversity of the input data, it can help to differentiate between inputs that are easier to classify or not, based on their ambiguity inside the training data domain. For example, given an image of a hand written four that is very similar to a nine, its prediction will have a higher aleatoric uncertainty than a clear image of the number one.

In the remaining sections of this paper we present our implementation of Bayesian neural networks using variational inference and our confidence measure formulation (2), we briefly analyze the most relevant alternative to our approach (3) and we show our results over multiple data sets belonging to different domains (4). Finally, we discuss the results and the advantages of our approach (5).

2 Methods

The idea behind Bayesian modeling is to consider all possible values for the parameters of a model when making predictions. In order to do so, the parameters are treated as random variables whose distribution is such that the most likely values are also the most probable ones.

The distribution of the model parameters, conditioned on the training data, is called posterior distribution and is defined using Bayes theorem:

$$p(\mathbf{w}|\mathcal{D}) = \frac{p(\mathcal{D}|\mathbf{w})p(\mathbf{w})}{p(\mathcal{D})} \tag{1}$$

where $\mathbf{w}$ is the set of model parameters and $\mathcal{D}$ is the training set. Its computation requires the definition of a likelihood function $p(\mathcal{D}|\mathbf{w})$, a prior density $p(\mathbf{w})$, and the computation of the marginal likelihood $p(\mathcal{D})$ that is unfeasible for complex models such as neural networks.

Mean Field Variational Inference

Variational inference allows to avoid computing the marginal likelihood by directly approximating the posterior distribution with a simpler one [Jordan et al., 1999]. In order to do so, it is necessary to minimize the Kullback–Leibler (KL) divergence between the proposed distribution and the posterior. The KL divergence is defined as follows:

$$KL\{q(\mathbf{w};\theta)||p(\mathbf{w}|\mathcal{D})\} = \int q(\mathbf{w};\theta)\log\frac{q(\mathbf{w};\theta)}{p(\mathbf{w}|\mathcal{D})}d\mathbf{w} \tag{2}$$

where θ is the set of variational parameters describing the proposed distribution q of the model's parameters $\mathbf{w}$. Since the posterior distribution is not known, we need to define a different objective to minimize the KL divergence. Such objective function is called Evidence Lower Bound (ELBO) and it is defined as follows:

$$ELBO = \mathbb{E}_{q(\mathbf{w};\theta)}\{\log p(\mathcal{D}|\mathbf{w})\} - KL\{q(\mathbf{w};\theta)||p(\mathbf{w})\} \tag{3}$$

Maximizing this lower bound with respect to the variational parameters θ of $q(\mathbf{w};\theta)$ provides a value as close as possible to the logarithm of the marginal likelihood and it is equivalent to minimizing the initial KL divergence between $q(\mathbf{w})$ and $p(\mathbf{w}|\mathcal{D})$ as proved in Appendix A. Variational inference turns the integration problem into an optimization one: maximizing the ELBO as a function of the variational parameters so that the proposed distribution fits the posterior [Hennig, 2011].

We approximated the posterior with a multivariate Gaussian distribution. Moreover, in order to simplify the optimization process, we used the mean field approximation that allows to factorize the approximating distribution as follows:

$$q(\mathbf{w};\theta) = \prod_{i=1}^{M} q(w_i;\theta_i) = \prod_{i=1}^{M} N(\mu_i,\sigma_i) \tag{4}$$

By using such approach, the number of latent variables to optimize is two times the number of parameters of the original model, in fact, for each weight w_i of the neural network, we optimize a univariate Gaussian parameterized by μ_i and σ_i. If we were to model the variational parameters as if they were not independent, and therefore consider also the covariance of the joint distribution of the parameters, this would correspond to a quadratic increase $(W(W+3)/2$, where W is the number of weights of the neural network) of the parameters to optimize which becomes prohibitive for complex models such as neural networks.

Bayesian Neural Network Training

Standard neural networks are usually trained by minimizing the cross entropy between the predictions and the classification targets. The optimization of the weights of the neural network is made by using the backpropagation algorithm. In order to be able to apply this algorithm to the variational parameters of a Bayesian neural network, it is necessary to separate the deterministic and the stochastic components of the weights, which now are random variables. This process takes the name of local reparameterization trick [Kingma et al., 2015] and its details are discussed in Appendix B.

Moreover, the loss function of the Bayesian neural network becomes the variational objective, i.e. the negative ELBO, where the likelihood can be divided in the sum of the contributions of all the individual data points in the dataset [Hoffman et al., 2013, Mandt et al., 2016, Mandt et al., 2017] and it is possible to employ a minibatch approach [Graves, 2011, Blundell et al., 2015]. The loss function can finally be defined in the following way:

$$f(\mathcal{D}_i,\theta) = -\mathbb{E}_{q(\mathbf{w};\theta)}\{\log p(\mathcal{D}_i|\mathbf{w})\} + \beta KL\{q(\mathbf{w};\theta)||p(\mathbf{w})\} \tag{5}$$

where $\mathcal{D}_i$ represents the *i*-th mini-batch, the log-likelihood log $p(\mathcal{D}_i|\mathbf{w})$ is the previously employed cross entropy function and $\beta = 1/M$ is the scaling factor of the KL divergence due to the minibatch approach, where *M* is the number of mini-batches composing the dataset. The prior distribution is a scale mixture prior composed of two Gaussian distributions as discussed in Appendix C.

Predictive Uncertainty

Predictions are made by weighting the predictions made with every parameter setting using the corresponding probability, that is the posterior probability. In order to approximate this integral, we use Monte Carlo samples from the approximating distribution q:

$$p(t|x,\mathcal{D}) = \int p(t|x,\mathbf{w})p(\mathbf{w}|\mathcal{D})d\mathbf{w} \approx \frac{1}{T}\sum_{t=1}^{T} p(t|x,\mathbf{w}_{(t)}), \quad \mathbf{w}_{(t)} \sim q(\mathbf{w};\theta) \tag{6}$$

where T is the number of output samples taken. Predictive uncertainty can be defined as the covariance of the predictive distribution. Since this measure does not consider both the epistemic and aleatoric components of uncertainty we extended the definition of uncertainty in the following way [Kwon et al., 2018, Kendall and Gal, 2017]:

$$\underbrace{\frac{1}{T}\sum_{t=1}^{T}\mathrm{diag}(\hat{p}_t) - \hat{p}_t^{\top}\hat{p}_t}_{\text{aleatoric}} + \underbrace{\frac{1}{T}\sum_{t=1}^{T}(\hat{p}_t - \bar{p})^{\top}(\hat{p}_t - \bar{p})}_{\text{epistemic}} \tag{7}$$

where $\bar{p} = \frac{1}{T}\sum_{t=1}^{T} \hat{p}_t$ and $\hat{p}_t$ is the output of the neural network after the softmax function has been applied. The resulting matrix contains information about the variance of each class on the diagonal as well as the covariance between different classes on the other elements. The latter components are not significant and it is hard to extract useful information from them, therefore we only considered the diagonal. Furthermore, we only considered the variance of the predicted class because of its ease of interpretation: a high predictive variance corresponds directly to high uncertainty for the prediction of that specific label. Therefore the predictive uncertainty can be computed in a simplified way:

$$u = \underbrace{\frac{1}{T}\sum_{t=1}^{T} \hat{p}_{t,i} - \hat{p}_{t,i}^2}_{\text{aleatoric}} + \underbrace{\frac{1}{T}\sum_{t=1}^{T} (\hat{p}_{t,i} - \bar{p}_i)^2}_{\text{epistemic}} \tag{8}$$

where $\hat{p}_{t,i}$ and $\bar{p}_i$ correspond to the i^{th} elements of the vectors $\hat{p}_t$ and $\bar{p}$, and i is the index of the predicted class. We finally transformed such uncertainty measure into a more intuitive Bayesian confidence score $c = 1 - 2\sqrt{u}$ which is also easier to compare to the standard neural networks confidence score.

3 Related work

There exist many solutions to find the posterior distribution of complex models such as Bayesian neural networks. Among them, it is worth citing the work regarding the Laplace approximation [Ritter et al., 2018], that approximates the posterior with a Gaussian distribution similarly to variational inference, and the Markov Chain Monte Carlo methods [Neal et al., 2011], that approximate the posterior by directly sampling from it. The most relevant of such techniques is dropout [Hinton et al., 2012], a regularization technique that, when active both at training and test time, has been proved to be equivalent to the approximation of a Bayesian neural network [Gal and Ghahramani, 2016].

Dropout has been extensively used to approximate Bayesian neural networks because of its ease of implementation and retrieval of predictive uncertainty estimates. Moreover, dropout has also shorter training and prediction times when compared to the previously mentioned approaches. For these reasons, dropout based Bayesian uncertainty measures have also been used to perform biomedical image analysis and prediction [Dürr et al., 2018, Leibig et al., 2017]. However recent work has exposed the main limitations of such approach, mainly related to the use of improper priors and the correctness of the variational objective [Hron et al., 2018].

4 Results

In the results we analyse different datasets and we use uncertainty estimates to improve the predictive power by discarding wrong predictions and identify out-of-distribution input samples. The architectures of the neural networks and the corresponding training hyperparameters are illustrated in Appendix E.

4.1 Benchmark

In this section, we examine our method against a well understood benchmark dataset for image analysis (MNIST, [LeCun and Cortes, 1998]), as well as uncertainty prediction against out-of-distribution data from a closely related dataset (EMNIST, [Cohen et al., 2017]).

4.1.1 Handwritten digits: MNIST

We first trained a Bayesian convolutional neural network using the MNIST training set and tested the model on the corresponding test set with $T = 100$ output samples (Appendix D). When we analyzed the output samples individually, they all showed good classification accuracy with small fluctuations (Figure1A). As demonstrated by predictions for 200 example images (Figure 1B), some images have a consistent prediction, while others produce more than one classification result over the range of samples, which indicates low confidence. For each image, we computed the Bayesian prediction by taking the average of the output samples as explained in the Methods section and illustrated in Figure 1C.

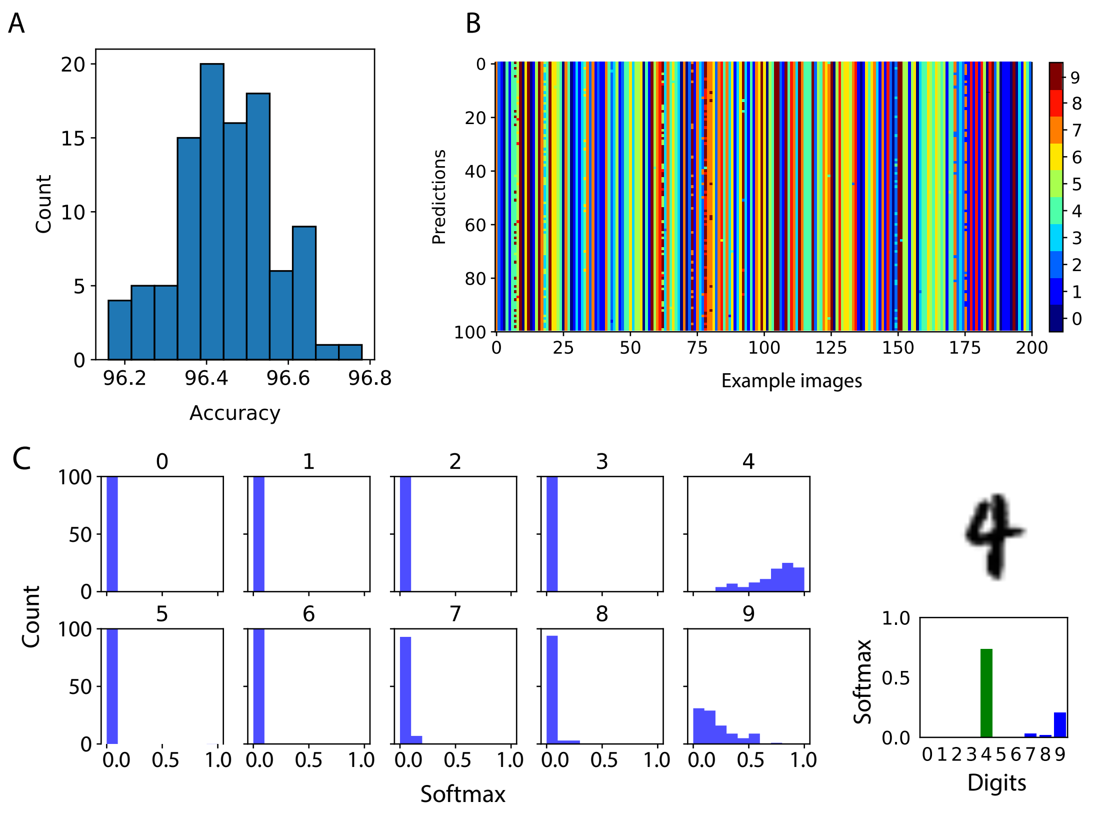

Figure 1: Bayesian neural network analysis of MNIST. A) Model accuracy over 100 samples. B) Predictions from 100 samples over 200 example images; color indicates predicted class. C) For the given example image '4', softmax score distributions are plotted for each of the 10 digits where the X axis is the value of softmax scores and the Y axis is the count. All 10 digits are then summarized into one plot where the X axis is the digits and the Y axis is the average over the 100 output samples. The digit with the highest softmax is reported as the predicted class, and colored green if correct, red if incorrect.

We then compared the Bayesian prediction results with a standard neural network trained on the same MNIST data. As shown in Figure 2A, confidence scores from the standard neural network tend to be close to 1 and there is no differentiation between correct and wrong predictions. In contrast, Bayesian confidence scores are high for correct predictions and low for incorrect ones. Indeed it is possible to improve the overall accuracy by increasing the confidence threshold and retaining only high-confidence predictions (Figure 2B). We also plotted the confidence scores from the Bayesian neural network and the standard neural network against each other for each test image (Figure 2C). By examining individual images, we see that images with high Bayesian confidence are the canonical digits while images with low Bayesian confidence correspond to corrupted or ambiguous observations. The standard neural network however is not able to distinguish them.

4.1.2 Out of distribution: EMNIST

The EMNIST dataset is designed with the same image format as MNIST, but it expands to include hand-written letters, both upper and lower case, for a total of 62 classes (10 digits, 26 lowercase, 26 uppercase). In order to validate the capability of the model to identify out-of-distribution samples — that is, images that cannot be labeled with any of the possible classes — we performed predictions over the EMNIST dataset with the Bayesian neural network model trained on the MNIST dataset.

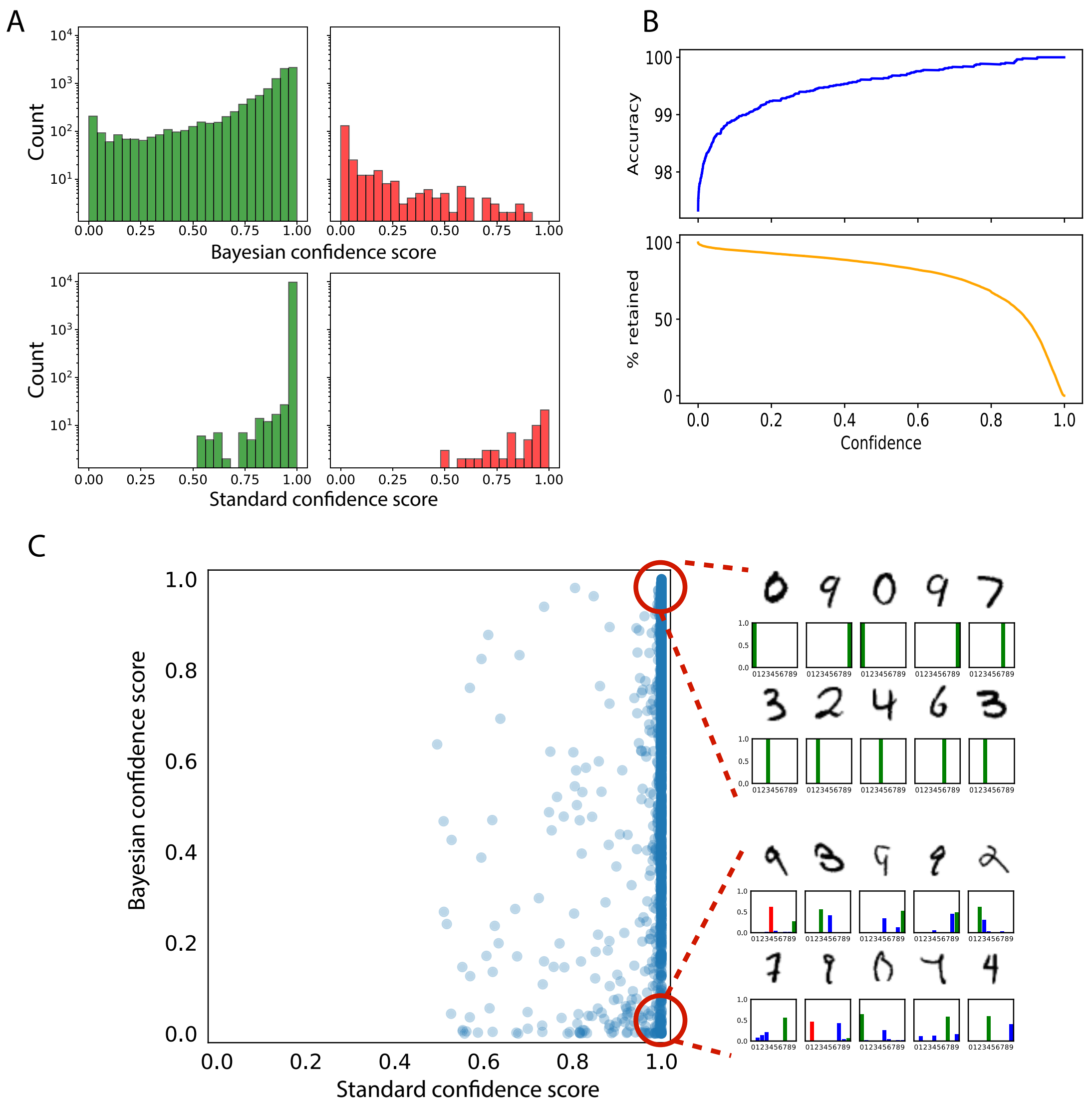

Figure 2: Comparison of Bayesian neural network and standard neural network predictions of the MNIST test set. A) Distribution of confidence scores for correct and incorrect predictions of both Bayesian neural network and standard neural network. B) Increasing confidence cutoff increases accuracy while decreases the percentage of retained images for Bayesian neural network analysis. C) A scatter plot with the X axis representing the confidence score from the standard neural network and the Y axis representing the confidence score from the Bayesian neural network. Example images from the top right and bottom right corners are shown, together with their corresponding Bayesian neural network predictions (as described in Figure 1C).

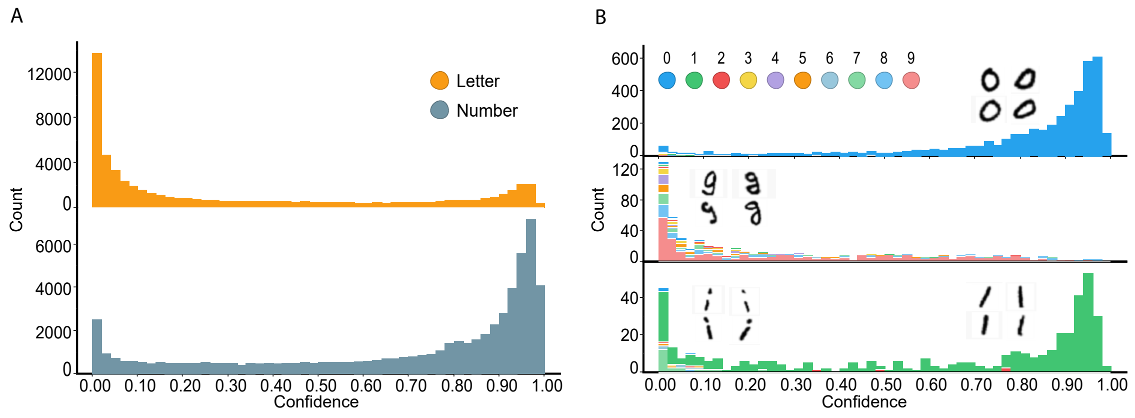

Figure 3: Applying Bayesian neural networks trained with MNIST on EMNIST. A) Confidence distribution for letters and numbers in EMNIST. B) Confidence distribution for three example letters, "O', 'g' and 'i'. Colors represent the predicted digits.

As shown in Figure 3A, the model predicts most numbers with high confidence, while predicts letters with low confidence as they are out-distribution-samples compared the MNIST training data. We examine the confidence distribution by each of the 62 classes and plot here three representative examples, 'O', 'g', and 'i' (Figure 3B). As expected, the letter 'O' is predicted as 0 with high confidence. On the other hand, as letter 'g' does not resemble any of the digits, the BNN model correctly predicts with low confidence. For letter 'i', interestingly the confidence scores show a bimodal distribution. After manually checking individual images, we realize that some 'i's are very similar to the number 1 which are predicted with high confidence, while other 'i's are written with a dot on top and the BNN model considers them as out-of-distribution samples with low confidence.

4.2 Cellular microscopy: BBBC021

As part of the Broad Bioimage Benchmark Collection (BBBC), the BBBC021 dataset are microscopy images of human MCF-7 breast cancer cells treated with 113 compounds over eight concentrations and labeled with fluorescent markers for DNA, F-actin, and β-tubulin [Ljosa et al., 2012, Caie et al., 2010]. A subset of BBBC021 with 38 compounds is annotated with a known mechanism of action (MoA). The MoA annotation comes both from visual inspection by experts as well as scientific literature, using a course-grained set of 12 MoAs, with each MoA containing multiple compounds and concentrations. This subset with ground truth annotation is widely used to benchmark diverse analysis methods [Ljosa et al., 2013, Kandaswamy et al., 2016, Godinez et al., 2017, Ando et al., 2017]. We developed the Bayesian approach as described above, with the same Multi-scale Convolutional architecture neural network (MCNN) architecture we previous designed [Godinez et al., 2017]. For validation, we adopted the rigorous leave-one-compound-out process, where in each session all except one compound was used for training and the hold-out compound was used for validation. Thus no single compound was present in both training and test data to prevent leakage. We trained the Bayesian neural network 38 times so each compound is predicted and evaluated once.

Figure 4A illustrates the predictive confidence for all hold-out compounds. The false prediction distribution has a peak on the low confidence side and the correct prediction distribution has a peak on the high confidence side. As a consequence, one can increase the overall accuracy by only retaining predictions with higher confidence (Figure 4B). We further examine the behavior of the BNN predictions for each of the 12 MoAs. Figure 4C shows the three compounds under the MoA 'Protein synthesis', where most images are predicted correctly and the false predictions are with low confidence.

The effects of compound treatment are complex due to how compounds interact with one or multiple protein targets and how these effects are reflected on cell morphology labeled with fluorescent markers. The MoA labels, although the best to our knowledge based on visual inspection and previous experiments, are often noisy and sometimes mistaken. In the MoA class 'Microtubule destabilizers', we have an interesting observation (Figure 5A). Most images from all three compounds within this MoA class, nocodazole, demecolcine and

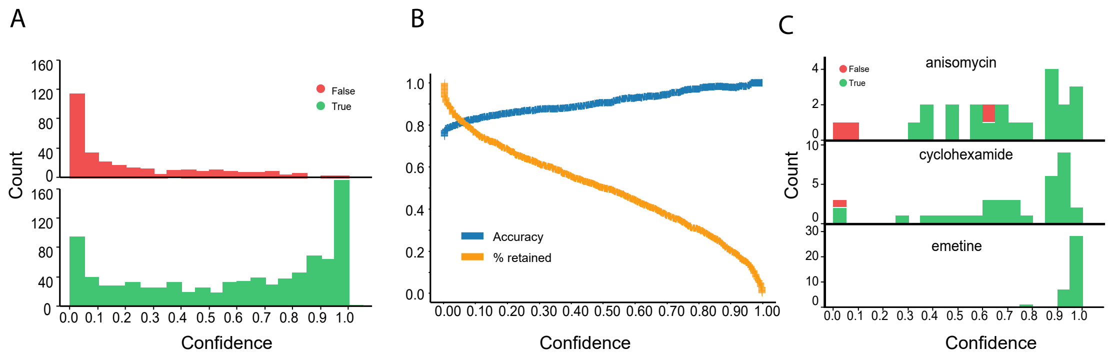

Figure 4: Bayesian neural network prediction in BBBC021. A) Distribution of confidence for all images correctly or incorrectly predicted. B) Increasing confidence cutoff increases accuracy while decreases the percentage of retained images. C) For the 'Protein synthesis' MoA, most images from all three compounds are correctly predicted.

vincristine, are predicted correctly. A small number of images are predicted incorrectly with low confidence. However, the fourth compound, colchicine, is predicted wrong for all images with a range of confidence scores. This seems to be inconsistent with how we have been interpreting confidence scores. However, when one manually examines the images, it is apparent that images of colchicine look different compared with the other three compounds (Figure 5B). This observation is consistent with the previous unsupervised approach applied on BBBC021 [Godinez et al., 2018] and indicates the noise in the ground truth annotation.

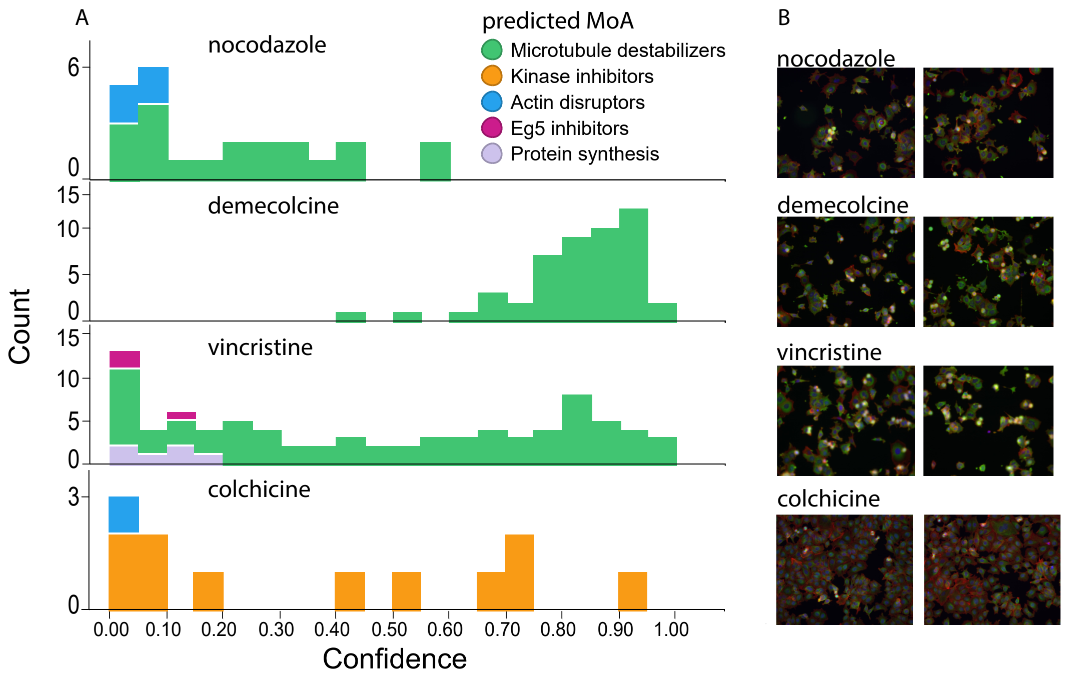

Figure 5: Uncertainty analysis for the 'Microtubule destabilizers' MoA. A) Distribution of confidence for samples from all four compounds annotated as 'Microtubule destabilizers'. Colors represent different predicted MoAs. All images from colchicine are predicted wrong while images from the other three compounds are mostly predicted correctly. B) Example images corresponding to each of the four compounds. Images from colchicine are notably different from the rest.

5 Discussion

The performance of deep neural networks in computer vision tasks has been explored for biological research and drug development. Compared with natural images, biomedical images have various challenges, such as noisy labels and out-of-distribution samples.

To address these challenges, we have implemented Bayesian neural networks with variational inference. By calculating a confidence score based on the predictive variance, one can estimate uncertainty of the predicted class labels. We first tested the Bayesian neural network approach in MNIST and EMNIST as proof-of-concept, and then applied on a biomedical image dataset where cells were treated with diverse drugs. With uncertainty analysis, we showed that one can achieve better accuracy by only relying on predictions with high confidence, and identified images as out-of-distribution samples although they shared the same ground-truth labels. We believe this Bayesian neural network approach has added value compared to standard neural networks in biomedical image classification.

Ground-truth labels for biomedical images are often impossible or prohibitively expensive to generate, which is why the field has moved towards unsupervised approaches. We envision future directions on developing Bayesian neural networks for clustering or image retrieval without any labels.

Acknowledgement

GD would like to acknowledge Prof. Maurizio Filippone (Eurecom, Data Science Department) for the supervision of GD's Master thesis[2] from which this work derives.

References

[Ando et al., 2017] Ando, D. M., McLean, C. Y., and Berndl, M. (2017). Improving phenotypic measurements in high-content imaging screens. *bioRxiv*.

[Benítez et al., 1997] Benítez, J. M., Castro, J. L., and Requena, I. (1997). Are artificial neural networks black boxes? *IEEE Transactions on neural networks*, 8(5):1156–1164.

[Blundell et al., 2015] Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. (2015). Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*.

[Caie et al., 2010] Caie, P. D., Walls, R. E., Ingleston-Orme, A., Daya, S., Houslay, T., Eagle, R., Roberts, M. E., and Carragher, N. O. (2010). High-content phenotypic profiling of drug response signatures across distinct cancer cells. *Molecular Cancer Therapeutics*, 9(6):1913–1926.

[Cohen et al., 2017] Cohen, G., Afshar, S., Tapson, J., and van Schaik, A. (2017). EMNIST: an extension of MNIST to handwritten letters. *arXiv preprint arXiv:1702.05373*.

[Der Kiureghian and Ditlevsen, 2009] Der Kiureghian, A. and Ditlevsen, O. (2009). Aleatory or epistemic? does it matter? *Structural Safety*, 31(2):105–112.

[Duch, 2003] Duch, W. (2003). Coloring black boxes: visualization of neural network decisions. In *Neural Networks, 2003. Proceedings of the International Joint Conference on*, volume 3, pages 1735–1740. IEEE.

[Dürr et al., 2018] Dürr, O., Murina, E., Siegismund, D., Tolkachev, V., Steigele, S., and Sick, B. (2018). Know when you don't know: A robust deep learning approach in the presence of unknown phenotypes. *Assay and drug development technologies*, 16(6):343–349.

[Gal and Ghahramani, 2016] Gal, Y. and Ghahramani, Z. (2016). Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059.

[Ghahramani, 2015] Ghahramani, Z. (2015). Probabilistic machine learning and artificial intelligence. *Nature*, 521(7553):452.

[Godinez et al., 2017] Godinez, W. J., Hossain, I., Lazic, S. E., Davies, J. W., and Zhang, X. (2017). A multi-scale convolutional neural network for phenotyping high-content cellular images. *Bioinformatics*, 33(13):2010–2019.

[2]https://webthesis.biblio.polito.it/10920/1/tesi.pdf

[Godinez et al., 2018] Godinez, W. J., Hossain, I., and Zhang, X. (2018). Unsupervised phenotypic analysis of cellular images with multi-scale convolutional neural networks. *bioRxiv*.

[Graves, 2011] Graves, A. (2011). Practical variational inference for neural networks. In *Advances in neural information processing systems*, pages 2348–2356.

[Hennig, 2011] Hennig, P. (2011). *Approximate inference in graphical models*. PhD thesis, University of Cambridge.

[Hinton et al., 2012] Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

[Hinton and Van Camp, 1993] Hinton, G. E. and Van Camp, D. (1993). Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the sixth annual conference on Computational learning theory*, pages 5–13. ACM.

[Hoffman et al., 2013] Hoffman, M. D., Blei, D. M., Wang, C., and Paisley, J. (2013). Stochastic variational inference. *The Journal of Machine Learning Research*, 14(1):1303–1347.

[Hron et al., 2018] Hron, J., Matthews, A. G. d. G., and Ghahramani, Z. (2018). Variational bayesian dropout: pitfalls and fixes. *arXiv preprint arXiv:1807.01969*.

[Jordan et al., 1999] Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., and Saul, L. K. (1999). An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233.

[Kandaswamy et al., 2016] Kandaswamy, C., Silva, L. M., Alexandre, L. A., and Santos, J. M. (2016). High-Content Analysis of Breast Cancer Using Single-Cell Deep Transfer Learning. *Journal of biomolecular screening*, 21(3):252–9.

[Kendall and Gal, 2017] Kendall, A. and Gal, Y. (2017). What uncertainties do we need in bayesian deep learning for computer vision? In *Advances in neural information processing systems*, pages 5574–5584.

[Kingma and Ba, 2014] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[Kingma et al., 2015] Kingma, D. P., Salimans, T., and Welling, M. (2015). Variational dropout and the local reparameterization trick. In *Advances in Neural Information Processing Systems*, pages 2575–2583.

[Kingma and Welling, 2013] Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

[Kwon et al., 2018] Kwon, Y., Won, J.-H., Joon Kim, B., and Paik, M. (2018). Uncertainty quantification using bayesian neural networks in classification: Application to ischemic stroke lesion segmentation. In *Medical Imaging with Deep Learning*.

[LeCun et al., 2015] LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436.

[LeCun and Cortes, 1998] LeCun, Y. and Cortes, C. (1998). The MNIST database of handwritten digits. *http://yann.lecun.com/exdb/mnist/*.

[Leibig et al., 2017] Leibig, C., Allken, V., Ayhan, M. S., Berens, P., and Wahl, S. (2017). Leveraging uncertainty information from deep neural networks for disease detection. *Scientific reports*, 7(1):17816.

[Ljosa et al., 2013] Ljosa, V., Caie, P. D., ter Horst, R., Sokolnicki, K. L., Jenkins, E. L., Daya, S., Roberts, M. E., Jones, T. R., Singh, S., Genovesio, A., Clemons, P. A., Carragher, N. O., and Carpenter, A. E. (2013). Comparison of Methods for Image-Based Profiling of Cellular Morphological Responses to Small-Molecule Treatment. *Journal of Biomolecular Screening*, 18(10):1321–1329.

[Ljosa et al., 2012] Ljosa, V., Sokolnicki, K. L., and Carpenter, A. E. (2012). Annotated high-throughput microscopy image sets for validation.

[Lundberg and Lee, 2017] Lundberg, S. M. and Lee, S.-I. (2017). A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, pages 4765–4774.

[Mandt et al., 2016] Mandt, S., Hoffman, M., and Blei, D. (2016). A variational analysis of stochastic gradient algorithms. In *International Conference on Machine Learning*, pages 354–363.

[Mandt et al., 2017] Mandt, S., Hoffman, M. D., and Blei, D. M. (2017). Stochastic gradient descent as approximate bayesian inference. *The Journal of Machine Learning Research*, 18(1):4873–4907.

[Mitchell and Beauchamp, 1988] Mitchell, T. J. and Beauchamp, J. J. (1988). Bayesian variable selection in linear regression. *Journal of the American Statistical Association*, 83(404):1023–1032.

[Neal et al., 2011] Neal, R. M. et al. (2011). MCMC using Hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 2(11):2.

[Ranganath et al., 2014] Ranganath, R., Gerrish, S., and Blei, D. (2014). Black box variational inference. In *Artificial Intelligence and Statistics*, pages 814–822.

[Rezende et al., 2014] Rezende, D. J., Mohamed, S., and Wierstra, D. (2014). Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*.

[Ritter et al., 2018] Ritter, H., Botev, A., and Barber, D. (2018). A scalable laplace approximation for neural networks. In *International Conference on Learning Representations*.

[Shrikumar et al., 2017] Shrikumar, A., Greenside, P., and Kundaje, A. (2017). Learning important features through propagating activation differences. *arXiv preprint arXiv:1704.02685*.

Appendix

A Evidence Lower Bound derivation

Variational inference employs the exclusive version of the KL divergence. Since the posterior distribution is not known, a different objective needs to be defined starting from the logarithm of the marginal likelihood of the model:

$$\log p(\mathcal{D}) = \log \int p(\mathcal{D}, \mathbf{w}) d\mathbf{w}$$

The computation of the marginal likelihood is the core issue of the Bayes theorem, in order to proceed we use an auxiliary function which corresponds to the proposal for the posterior approximation $q(\mathbf{w}|\theta)$ that we write as $q(\mathbf{w})$ to simplify the notation:

$$\log p(\mathcal{D}) = \log \int q(\mathbf{w}) \frac{p(\mathcal{D}, \mathbf{w})}{q(\mathbf{w})} d\mathbf{w} = \log \mathbb{E}_{q(\mathbf{w})} \left\{ \frac{p(\mathcal{D}, \mathbf{w})}{q(\mathbf{w})} \right\}$$

Then it is needed to bring the logarithm inside the expectation which can be done by applying the Jensen's inequality:

$$\log \mathbb{E}_{q(\mathbf{w})} \left\{ \frac{p(\mathcal{D}, \mathbf{w})}{q(\mathbf{w})} \right\} \geq \mathbb{E}_{q(\mathbf{w})} \left\{ \log \frac{p(\mathcal{D}, \mathbf{w})}{q(\mathbf{w})} \right\}$$

Since we are now using an inequality, its right term is a lower bound of the logarithm of the marginal likelihood, also called model evidence, hence the name Evidence Lower Bound (ELBO). The ELBO can now be reformulated in the following way:

$$\begin{aligned}
\mathbb{E}_{q(\mathbf{w})} \left\{ \log \frac{p(\mathcal{D}, \mathbf{w})}{q(\mathbf{w})} \right\} = \int q(\mathbf{w}) \log \frac{p(\mathcal{D}, \mathbf{w})}{q(\mathbf{w})} d\mathbf{w} = \int q(\mathbf{w}) \log \frac{p(\mathcal{D}|\mathbf{w}) p(\mathbf{w})}{q(\mathbf{w})} d\mathbf{w} = \\
= \int q(\mathbf{w}) \log p(\mathcal{D}|\mathbf{w}) d\mathbf{w} - \int q(\mathbf{w}) \log \frac{p(\mathbf{w})}{q(\mathbf{w})} d\mathbf{w} = \\
= \mathbb{E}_{q(\mathbf{w})} \{\log p(\mathcal{D}|\mathbf{w})\} - KL\{q(\mathbf{w}) || p(\mathbf{w})\} = \mathcal{L}(\theta)
\end{aligned}$$

Maximizing this lower bound with respect to the variational parameters θ of $q(\mathbf{w}; \theta)$ provides a value as close as possible to the logarithm of the marginal likelihood and it is equivalent to minimizing the initial KL divergence between $q(\mathbf{w})$ and $p(\mathbf{w}|\mathcal{D})$:

$$\begin{aligned}
KL\{q(\mathbf{w}) || p(\mathbf{w}|\mathcal{D})\} = \int q(\mathbf{w}) \log \frac{q(\mathbf{w})}{p(\mathbf{w}|\mathcal{D})} d\mathbf{w} = \int q(\mathbf{w}) \log \frac{q(\mathbf{w}) p(\mathcal{D})}{p(\mathcal{D}|\mathbf{w}) p(\mathbf{w})} d\mathbf{w} = \\
= - \int q(\mathbf{w}) \log p(\mathcal{D}|\mathbf{w}) d\mathbf{w} + \int q(\mathbf{w}) \log \frac{q(\mathbf{w})}{p(\mathbf{w})} d\mathbf{w} + \int q(\mathbf{w}) \log p(\mathcal{D}) d\mathbf{w} =
\end{aligned}$$

$$= -\mathbb{E}_{q(\mathbf{w})}\{\log p(\mathcal{D}|\mathbf{w})\} + KL\{q(\mathbf{w})||p(\mathbf{w})\} + \log p(\mathcal{D})$$
$$KL\{q(\mathbf{w})||p(\mathbf{w}|\mathcal{D})\} = -\mathcal{L}(\theta) + \log p(\mathcal{D})$$

The marginal likelihood is constant, therefore maximizing the ELBO is equivalent to minimizing the KL divergence between the posterior and its approximation.

B Reparameterization of neural network weights

Bayesian neural networks weights are sampled from the posterior for each iteration of the training process, therefore it is not possible to optimize directly the variational parameters because the updates do not persist across iterations.

A solution to this problem is the reparameterization trick [Kingma and Welling, 2013, Blundell et al., 2015, Rezende et al., 2014] which allows to write the weights of the neural network as a function of the variational parameters. Assuming a Gaussian posterior approximation, the weights can be reparameterized as follows:

$$w = \mu + \sigma\epsilon, \quad \epsilon \sim \mathcal{N}(0,1)$$

The trainable variational parameters μ and σ are now separated from the random variable ϵ and it is possible to optimize them.

The output of each layer of the Bayesian neural network depends on the value of ϵ, hence the computation of the predictions is a stochastic process depending on the expected value of the network's weights. This expectation can be approximated using Monte Carlo sampling and depends on the number of MC samples used: a higher number of samples usually leads to faster convergence but not necessarily to better optima.

A step of the final optimization process using one Monte Carlo sample can be summarized as follows:

1. Sample $\epsilon_i \sim \mathcal{N}(0,1)$ for each weight of the neural network;
2. Let $w_i = \mu_i + \sigma_i\epsilon_i$ for each weight of the neural network;
3. Let $f(\mathcal{D},\theta) = -\log p(\mathcal{D}|w) + \log \frac{q(w;\theta)}{p(w)}$, with $\theta = \{\mu, \sigma\}$;
4. Compute the gradient of f with respect to the variational parameters: $\nabla_\mu, \nabla_\sigma$;
5. Update the variational parameters with learning rate η:

$$\mu \leftarrow \mu - \eta\nabla_\mu, \quad \sigma \leftarrow \sigma - \eta\nabla_\sigma$$

Finally, since the update of σ could lead to impossible negative values, we reparameterized it with the *softmax* function using a parameter ρ that can assume any value on the real axis:

$$\sigma = \log(1 + exp(\rho))$$

The optimization process is stochastic and depends both on the number of data points used for each update of the parameters and the sampled values of the model parameters at each iteration. This second factor could increase considerably the variance of the gradients, leading to very slow convergence. A solution to decrease the variance of these gradients is the local reparameterization trick [Kingma et al., 2015] because it transforms the global noise ϵ of the model's weights into local noise of the output of each layer ϵ_z.

For a neural network's fully connected layer with a fully factorized Gaussian posterior distribution, the activation is a fully factorized normal distribution as well. A reparameterized sample is taken from the latter instead of the posterior approximation as follows:

$$z = \mu_z + \sigma_z\epsilon, \quad \epsilon \sim \mathcal{N}(0,1)$$

where μ_z and σ_z are the mean and variance of the output of the layer. They are computed using the layer input $\mathbf{X}$ and the variational parameters of the weights w and bias b:

$$\mu_z = \mathbf{X}\mu_w + \mu_b, \quad \sigma_z = \sqrt{\mathbf{X}^\top\mathbf{X}\sigma_w^2 + \sigma_b^2}$$

This technique has many advantages: it decreases the variance of the gradients and it is also more computationally efficient because it requires less random samples ϵ.

C Bayesian neural network prior distribution

A more detailed analysis of the variational objective leads to a possible interpretation of its two components [Hinton and Van Camp, 1993]:

$$\underbrace{\mathbb{E}_{q(\mathbf{w};\theta)}\{\log p(\mathcal{D}|\mathbf{w})\}}_{\text{model fit}} - \underbrace{KL\{q(\mathbf{w};\theta)||p(\mathbf{w})\}}_{\text{regularization}}$$

The expectation of the log likelihood is a model fit term, given samples from the posterior approximation q, it measures how well on average they fit the training data, a higher value means a better fit. The KL divergence between the posterior approximation and the prior distribution is not a distance metric, but it can be interpreted as a measure of how far are two distributions. In order to maximize the ELBO, this term needs to be minimized, therefore it is a regularization term because it aims to keep the solution as close as possible to the prior belief defined on the model parameters. Moreover, it can be proved that introducing a Gaussian prior on the model parameters is equivalent to L2 regularization while a Laplace prior yields L1 regularization [Blundell et al., 2015].

The choice for the prior distribution can be trivial for simple models such as linear regression because of the interpretability of model parameters. When this property is lost, e.g. logistic regression, and the number of parameters grows up to thousands or millions, e.g. neural networks, the choice of a good prior becomes harder but, nevertheless, fundamental to obtain good convergence of the algorithm and a good posterior approximation.

The regularization factor of a Gaussian prior is inversely proportional to the variance of the distribution. The variance of the weights distributions, i.e. the posterior, is updated to reduce the KL divergence, meaning that the posterior approximation tries to match the variance of the prior where it is possible. Therefore the variance needs to be small enough to allow a good regularization of the means but also big enough to let very uncertain weights get wider distributions. For this reason, a useful prior is the scale mixture distribution with two densities having both zero mean, and respectively very small and very big variances:

$$p(w) = \pi\mathcal{N}(0,\sigma_1) + (1-\pi)\mathcal{N}(0,\sigma_2), \quad \pi \in [0,1]$$

where π is used to assign different importance to the two components. The scale mixture prior can be used to approximate the spike and slab distribution [Mitchell and Beauchamp, 1988, Blundell et al., 2015] which is almost uniformly low with a spike on the zero. This shape causes many of the weights to have values very close to zero while still allowing greater values because it has heavier tails than a normal distribution.

The choice of a Gaussian prior, together with a fully factorized posterior approximation, has a computational advantage over more sophisticated priors because the KL divergence between two normal distributions can be computed analytically. On the other side, if a closed form solution is not available, it is possible to approximate the expectation using Monte Carlo samples [Ranganath et al., 2014]:

$$KL\{q(w)||p(w)\} = \mathbb{E}_{q(w)}\left\{\log\frac{q(w)}{p(w)}\right\} \approx \sum_{i=0}^{N}(\log q(w_{(i)}) - \log p(w_{(i)})), \quad w_{(i)} \sim q(w)$$

D Number of predictive samples

The predictive distribution is approximated using Monte Carlo samples, therefore, depending on the number of outputs averaged to compute the final prediction, the corresponding accuracy can be more or less precise. Figure 6 shows 100 accuracies per numbers of samples used to compute the output, and the corresponding median and variance per number of samples. As the number of samples increases we get a closer estimate of the real accuracy value and its variance decreases.

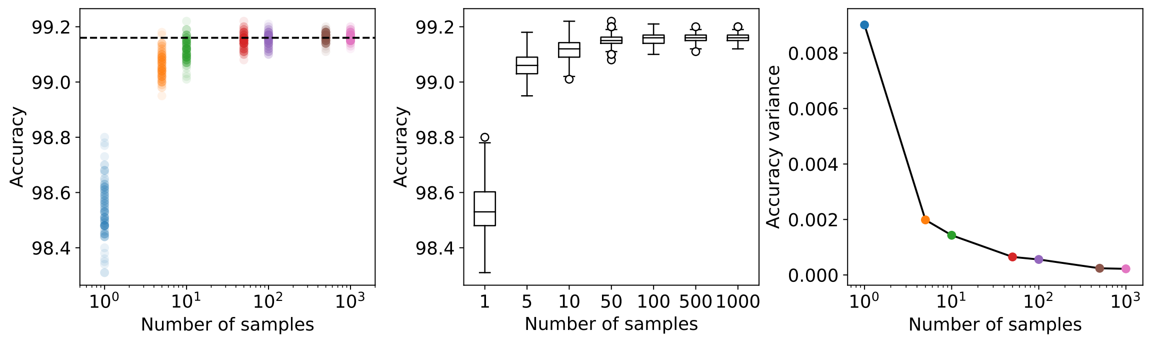

Figure 6: Scatter plot of 100 values of the accuracy of a Bayesian neural network per different number of samples (left), the corresponding boxplot (center) and the variance per number of samples (right).

Figure 6 data correspond to a Bayesian neural network trained on the MNIST dataset. We finally decided to compute predictions and accuracy using 100 samples because the corresponding estimate variance is smaller than 0.001 and the median is close enough to the median computed on 1000 samples, which is assumed to be a good estimate of the accuracy. The main advantage of using only 100 samples is the computational time required to make predictions: while computing 1000 outputs for a single image takes approximately 1.2 seconds, it only takes approximately 0.1 seconds to get 100.

E Neural networks architectures and training hyperparameters

The architectures used for the experiments are illustrated in Figure 7. We used the same architectures for both the standard and Bayesian case in order to be consistent when performing comparison. We performed the optimization of the neural networks parameters using the Adam optimizer [Kingma and Ba, 2014]. The training hyperparameters for all the architectures used are illustrated in Table 1.

In order to train the standard LeNet5 we used the cross entropy loss function and a learning rate schedule consisting of a single step at epoch 170, decreasing the learning rate by a factor of 10. Furthermore, we added weight decay with a factor of 5×10^{-3} to the loss function to regularize the training and give the model better generalization. It is important to underline that this is not the best model to train on the MNIST dataset and a wide number of further improvements could have been used (e.g. Dropout, Batch Normalization, Data augmentation). This implementation choice has been taken because the model is still able to perform very well and, since the dataset is contextually simple, further improvements could lead to higher accuracies but complicate the comparison with the Bayesian counterpart.

Given the different nature of Bayesian LeNet5 with respect to the previous one, it has been trained with a lower batch size and for more epochs, with a single learning rate step at 200 epochs. The loss function is the variational objective, namely the ELBO loss. No weight decay has been applied because the regularization is already imposed by the shape and parameters of the prior.

In order to work with the BBBC021 dataset we initially preprocessed the data. We normalizated the images by plate and channel by adjusting the pixels intensity p using extremely low and high values (0.03 and 99.97 percentile), and clipping the result within the range [0, 1] as follows:

$$\hat{p}_{xy} = \left. \frac{p_{xy} - p_{0.03\%}}{p_{99.97\%} - p_{0.03\%}} \right]_0^1 \tag{9}$$

Moreover, we excluded the negative control and truncated the microtubule stabilizers class to the median length of the classes because of the highly unbalanced count. By doing so the dataset is still unbalanced but the number of observations for each class is comparable. Finally, we adopted a weighted loss function when training the neural network. The weights of this loss were computed as the inverse of the number of observations for each class, so that all the classes have the same influence on the updates of the model parameters. We finally trained the neural network using a learning rate schedule consisting of a single step at epoch 60.

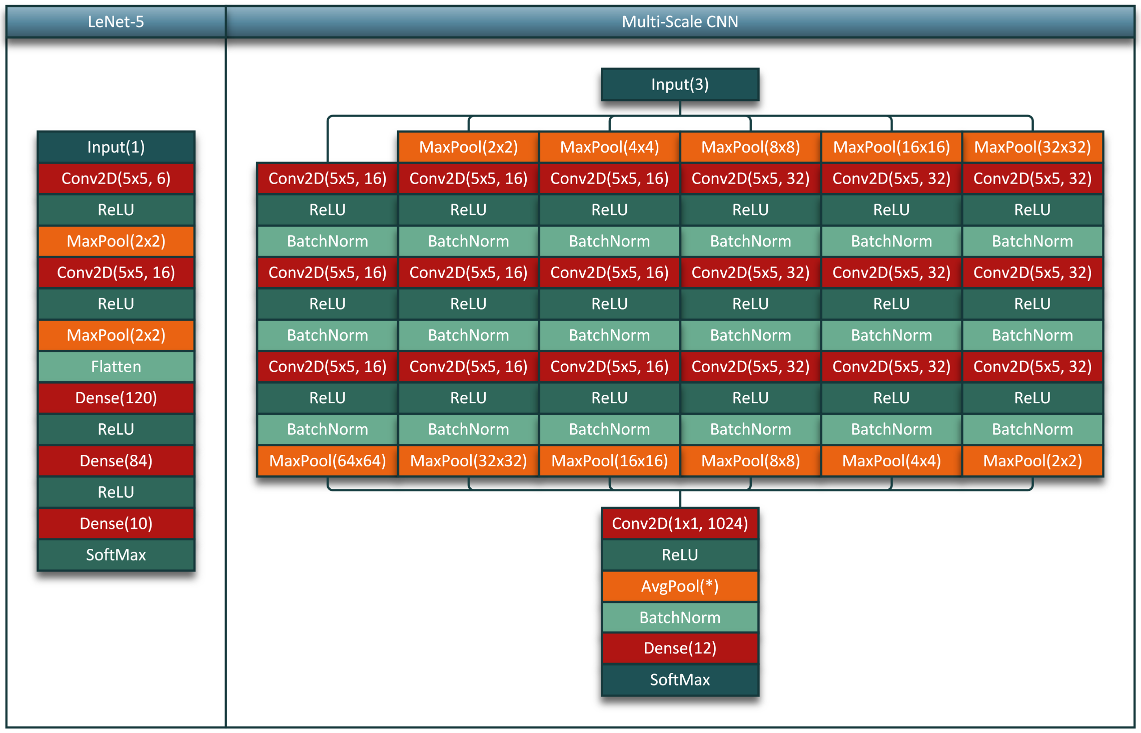

Figure 7: Neural networks architectures.

Neural network architecture	Dataset	# Epochs	Batch size	Learning rate
Standard LeNet5	MNIST	200	256	10^{-3}
Bayesian LeNet5	MNIST	300	32	10^{-3}
Bayesian MSCNN	BBBC021	80	20	10^{-2}

Table 1: Training hyperparameters of the different architectures used in the experiments.