Master of Science

Double Degree in Data Science and Software Engineering

Uncertainty modeling in deep learning

Variational inference for Bayesian neural networks

Academic supervisors:

Prof. Maurizio Filippone

Data Science Department, EURECOM

Prof. Elisa Ficarra

DAUIN, Polytechnic University of Turin

Author:

Giacomo Deodato

Industrial supervisors:

Xian Zhang

Senior Investigator I

Novartis Institutes for Biomedical Research

Christopher Ball

Scientific Technical Leader II

Novartis Institutes for Biomedical Research

April, 2019

Rapport de Thèse Professionnelle

Giacomo Deodato

Février, 2019

Modélisation de l'incertitude dans l'apprentissage profond

Inférence variationnelle pour les réseaux neuronaux Bayésiens

Société: **Novartis AG**, Novartis Institutes for Biomedical Research

Encadrant dans l'entreprise: **Xian Zhang**, CBT Data Science, NIBR

Encadrant académique: **Prof. Maurizio Filippone**

Filière: **Data Science and Engineering**

Cette thèse n'est pas confidentielle

EURECOM

I would like to dedicate this thesis to the
people I met who are passionate about
research and still look at new discoveries
with the eyes of a child.

A nonno Giacomo e nonno Michele

"SUMASAO!"

All the images and results presented in this work were produced by the author unless specified differently. They can be reproduced using the code in the thesis repository and the corresponding variational inference framework located at github.com/giacomodeodato.

Acknowledgements

I would like to acknowledge all of my supervisors, Christopher Ball, Elisa Ficarra, Maurizio Filippone and Xian Zhang, for their patience and attention.

I am profoundly grateful to all the people who gave me advise on my work and provided useful comments and discussion. In particular, I would like to thank Kurt Cutajar and Giovanni Palla. I would also like to thank Emilio Benenati, Ilario Gerloni and all my friends who continue to support me every day.

Lastly, I would like to thank my family for allowing me to achieve my goals and the ERASMUS PLUS program for funding my stay in France.

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. After an introduction to the concepts behind the Bayesian framework we study variational inference and investigate its advantages and limitations in approximating the posterior distribution of the weights of neural networks. In particular, we underline the importance of the choice of a good prior, we analyze performance and uncertainty of models using normal priors and scale mixture priors, and we discuss the optimization of the variational objective during the training of the model.

Furthermore, 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.

The results show that the two priors used lead to similar predictive distributions but different posteriors. In fact, the scale mixture prior induces a better regularization and sparsity of the weights. Moreover, the analysis of the predictive uncertainty shows that it is possible to isolate both wrong predictions and out-of-distribution input samples, that are corrupted observations or data belonging to different domains. The latter property of the predictive uncertainty is finally used in a biomedical use case consisting of the application of a Bayesian neural network to identify new Mechanism-Of-Action between compounds and cells.

In conclusion, our study highlights the opportunities and challenges of the application of Bayesian neural networks in the context of image analysis, and proposes some best practices to train such models employing variational inference.

Résumé

Au cours des dernières décennies, les modèles d'apprentissage profond ont rapidement gagné en popularité pour leur capacité à obtenir des performances de pointe dans différents contextes d'inférence. Les réseaux neuronaux profonds ont été appliqués à un nombre croissant de problèmes couvrant différents domaines d'application. Les nouvelles applications définissent un nouvel ensemble d'exigences qui transcendent les prédictions précises et dépendent des mesures d'incertitude.

Les objectifs de cette étude sont de mettre en œuvre des réseaux neuronaux Bayésiens et d'utiliser les estimations d'incertitude correspondantes pour effectuer des prévisions et analyser des ensembles de données. Après une introduction aux concepts à la base du cadre Bayésien, nous étudions l'inférence variationnelle et examinons ses avantages et ses limites pour l'approximation de la distribution à posteriori des poids des réseaux neuronaux. En particulier, nous soulignons l'importance du choix d'une bonne distribution à priori, nous analysons les performances et l'incertitude des modèles utilisant des distributions à priori normales et des mélanges de Gaussiennes, et nous discutons l'optimisation de l'objectif variationnel lors de l'entraînement du modèle.

En outre, nous identifions deux avantages principaux dans la modélisation de l'incertitude prédictive des réseaux neuronaux profonds effectuant des tâches de classification. Le premier est la possibilité d'écarter des prédictions très incertaines afin de pouvoir garantir une plus grande précision des prédictions restantes. La seconde est l'identification dans les données de formes inconnues qui correspondent à des valeurs aberrantes dans la représentation du modèle de la distribution des données d'apprentissage.

Les résultats montrent que les deux distributions à priori utilisées conduisent à des distributions prédictives similaires mais à des distributions à posteriori différentes. En fait, la distribution de mélange Gaussienne induit une meilleure régularisation et à des poids clairsemés. De plus, l'analyse de l'incertitude prédictive montre qu'il est possible d'isoler les prédictions erronées et les échantillons d'entrée hors distribution, qui sont des observations corrompues ou des données appartenant à différents domaines. La dernière propriété de l'incertitude prédictive est finalement utilisée dans un cas d'utilisation biomédicale consistant en l'application d'un réseau neuronal Bayésien pour identifier un nouveau mécanisme d'action entre des composés chimiques et des cellules.

En conclusion, notre étude met en évidence les opportunités et les défis de l'application des réseaux neuronaux Bayésiens dans le contexte de l'analyse d'images, et propose quelques meilleures pratiques pour former de tels modèles en utilisant l'inférence variationnelle.

Table of contents

Chapter 1

Introduction

Deep neural networks, a domain of modern machine learning research, 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). Deep neural networks consist in multiple layers of neurons, the basic unit of the model, that aim to optimize a set of algebraic operations in order to perform the prediction task with the highest degree of accuracy (Schmidhuber, 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; Lundberg and Lee, 2017; Shrikumar et al., 2017)) and their dependence on point estimates of their parameters. Despite their ability to outperform simpler models, for a variety of tasks and domain applications a single prediction score (i.e. the accuracy of the prediction) is not sufficient (Ghahramani, 2015). Modeling applications such as autonomous driving and 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.

Furthermore, uncertainty modeling can be useful also in less critical fields because it provides a way to get better insights about the data: more certain predictions correspond to cleaner data both from a technical and a contextual point of view. In the same way, uncertainty with respect to the model parameters is useful for model selection and to get a better understanding of how information is processed by the model. For these reasons, uncertainty modeling makes new room for improvement in the maturing field of deep learning

1.1 Neural networks limitations

Neural networks are models loosely inspired by biological neural networks because of the similarity between the basic computational unit of neural networks and a brain's neuron (Rosenblatt, 1958). They consist in a stack of layers of neurons, each layer utilizes a set of weights to perform a transformation on its input and further applies a non linear function (also called activation function) to the result. The output of a layer represents a new set of features that corresponds to the input for the following layer (Bishop, 2006).

A neural network with just one layer and a finite number of nodes can approximate a wide variety of continuous functions (Cybenko, 1989) hence neural networks are able to represent almost any mapping between a set of input features and the corresponding targets. Because of their excellent performance, neural networks represent the state-of-the-art for a large set of applications. However, after the weight optimization, it is not possible to provide a complete and exhaustive explanation of the weights values, because they are neither interpretable nor designed to produce any confidence measure with respect to the predictions.

The most common application of neural networks is supervised learning (LeCun et al., 2015): the parameters values are learned from a dataset containing a relatively large number of observations of the input-output mapping (Russell and Norvig, 2016). This approach can be used to perform regression as well as classification tasks. Neural networks can perform a variety of other tasks (e.g. unsupervised learning) exploiting also different architectures (i.e. the arrangement of neurons and layers), but the training usually follows the same process. For example, autoencoders are used to find a more compact representation of the input data by employing an architecture consisting of two main components: the encoder takes the input data and compresses them, producing the corresponding embedding, while the decoder reconstructs the original input data by transforming the embedding (Rumelhart et al., 1985). In order to train this model it is still necessary to provide a dataset of (input, output) couples, the only difference is the desired output corresponds to the initial input itself because the reconstruction error is used to train the parameters.

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, a generalization of the logistic function that transforms the output so that each value is in the range $[0,1]$ and they add up to 1 (Denker and Lecun, 1991). 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 the probability value of the output, this should not lead to think that this output feature is an appropriate measure of confidence as this would show that the model makes overconfident prediction most of time (Gal and Ghahramani, 2016). In fact, the softmax output is just the activation of the final layer of a neural network, and it does not capture the uncertainty of the model parameters or the data because the neural network is not designed to do so.

1.2 Uncertainty modeling

Uncertainty can be used to enhance the performance of standard neural networks as well as to better analyse the data. In particular, *predictive* uncertainty describes the confidence relative to the predictions and can be decomposed into model uncertainty, also called *epistemic* uncertainty, and *aleatoric* uncertainty, which depends on the noise of the observations (Der Kiureghian and Ditlevsen, 2009).

Epistemic uncertainty stems from model's parameters as well as the specific architecture of the model. Aleatoric uncertainty can be divided into an *homoscedastic* component and an *heteroscedastic* component, the former is constant for different inputs while the latter is input dependent. It is important to underline that 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 these reasons, uncertainty can be used for different tasks.

Aleatoric uncertainty is useful in big data scenarios where the epistemic component of uncertainty disappears, potentially leading the model to overconfident predictions. Moreover, 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. 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. A more complex example may be in the context of an autonomous driving system, aleatoric uncertainty can help the autopilot to distinguish between uncertain images of a country road where the margins of the street are not well defined, from images which are easier to interpret, e.g. a highway.

Epistemic uncertainty is useful when big data sets are not available because these are the scenarios that require to express a high uncertainty for the model parameters. Furthermore, it is key to understand *out-of-distribution* samples, observations that are different from the training data, e.g. an image of the letter R when the training data are digit images. This can be particularly useful for critical applications where it is not possible to train the model on the full distribution of input data. For instance, considering the previous autonomous driving example, it is not possible to train the neural network on all possible scenarios therefore the autopilot should notify the human driver to take control in the presence of uncertain images from the street. Epistemic uncertainty can also be used to find novel features in the input data.

For example, given a dataset of microscope images of cells cultured with different drugs, a classifier trained on a finite number of known interactions could spot new ones.

1.3 Remainder of the thesis

The remaining chapters of this thesis present Bayesian neural networks together with their applications using the predictive uncertainty and results on both toy datasets and real use cases. The second chapter is a coherent introduction to Bayesian modeling starting from linear regression and ending with the current issues and alternatives regarding Bayesian neural networks. In the third chapter we delve into the details of variational inference, its advantages and limitations, and discuss different practices in the implementation of Bayesian Neural Networks. The fourth chapter contains a benchmark of the Bayesian convolutional neural networks and the standard ones using a simple dataset. In particular, the weights uncertainty and the predictive uncertainty are analysed, showing the advantages of the probabilistic approach. Since this work has been produced in the Chemicals Biology and Therapeutics department of the Novartis Institutes for Biomedical Research (NIBR), the fifth chapter illustrates Biomedical use cases aiming at improving the drug discovery pipeline. Finally, the sixth chapter concludes with current research directions, open challenges and final remarks.

Chapter 2

Bayesian Modeling

In the introduction we presented the importance of uncertainty modeling for some critical applications as well as general machine learning problems. This chapter provides an introduction to the Bayesian approach starting from the simplest model, linear regression, and then generalizing to any kind of parameterized model. The main issue of Maximum Likelihood Estimates (MLE) models are analyzed to show the advantages of Bayesian modeling.

After this presentation of the main concepts behind the Bayesian framework, the alternative implementations of Bayesian neural networks are discussed. Furthermore, the link between Gaussian Processes and Bayesian neural networks is briefly analyzed and the chapter concludes with a summary of the properties of Bayesian models.

2.1 Linear Regression

Linear regression is the simplest model that can be used to perform supervised regression tasks because of the low number of parameters, their interpretability and ease of visualization. For these reasons it is a good starting point to introduce the concepts behind Bayesian modeling. Linear regression models compute the target t as a linear combination of the input features x_i:

$$t = w_0 + w_1 x_1 + \cdots + w_D x_D$$

where D is the number of features of each observation $\mathbf{x} = (x_1, ..., x_D)^\top$, and $\mathbf{w} = (w_0, ..., w_D)^\top$ is the vector of model parameters. By defining $x_0 = 1$ it is possible to express linear regression in the more compact form:

$$t = \mathbf{w}^\top \mathbf{x}$$

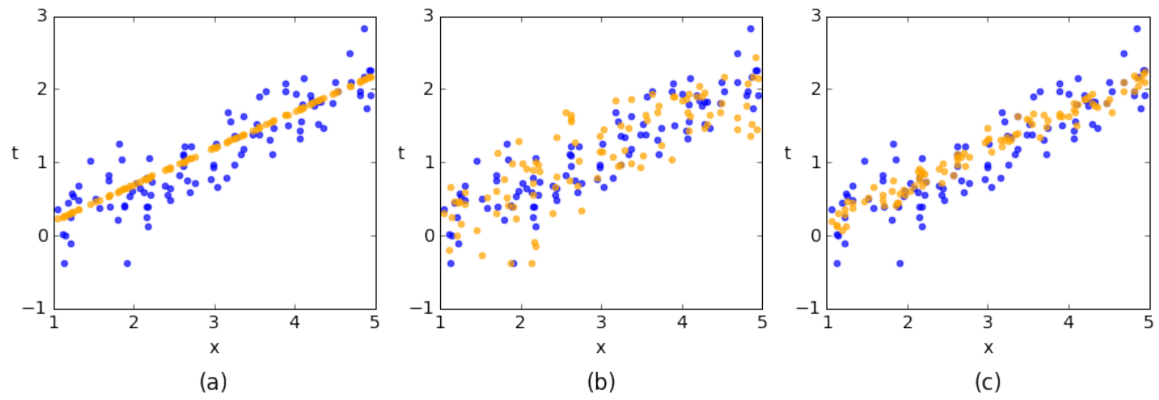


Fig. 2.1 Observations (blue) and model generated data (orange) with no variance (a), more variance (b) and less variance (c) than the observations.

2.1.1 Maximum Likelihood Estimates

An important property of the targets needs to be underlined, most real world data is corrupted or just distorted by noise coming from different sources, e.g. measurement tools in the collection process or random errors introduced in data preparation and processing (Nettleton et al., 2010).

The error of the model is strictly related to the variance of the observations and it can give a measure of confidence in the predictions. Figure 2.1 illustrates three different scenarios in which we try to generate the input data using the optimized model. In Figure 2.1(a) the variance of the inputs is not considered and thedata lie on the line $t = \mathbf{w}^\top \mathbf{x}$. Figure 2.1(b) and (c), instead, show wrong generated data obtained by adding random noise to the original values. Since the variance of the random noise is assumed it may result being either lower or higher than the real value, hence the model would be making overconfident or underconfident predictions.

For this reason it i s more accurate to model the target as a random variable so that it is possible to threat the noise variance as a parameter. In this particular scenario, it can be assumed that observations of t are perturbed with Gaussian noise ε with variance σ^2. Given a data set of N observations $(\mathbf{x}_n, t_n)$:

$$t_n = \mathbf{w}^\top \mathbf{x}_n + \varepsilon_n, \quad \varepsilon_n \sim \mathcal{N}(0, \sigma^2)$$

$$p(t|\mathbf{x}_n, \mathbf{w}, \sigma^2) = \mathcal{N}(\mathbf{w}^\top \mathbf{x}_n, \sigma^2)$$

Since now t is a distribution, it is possible to define the likelihood as the value obtained for $p(t = t_n|\mathbf{x}_n, \mathbf{w}, \sigma^2)$. The likelihood is a measure of how likely is the model with parameters $\mathbf{w}$

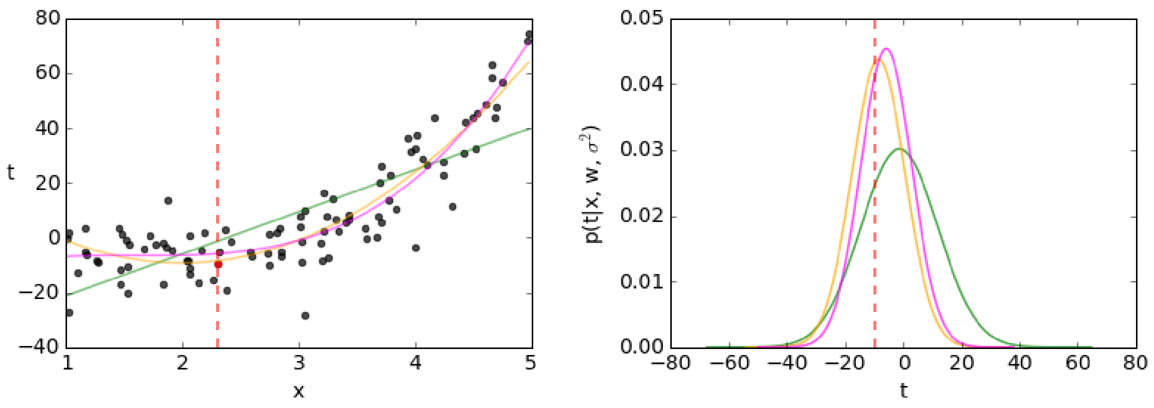


Fig. 2.2 Polynomials of 1st (green), 2nd (orange) and 3rd (magenta) degree fitting artificially generated data (left) and likelihood of the red data point for the three different models (right).

and σ^2 to generate the target t_n given the input $\mathbf{x}_n$. Figure 2.2 illustrates three polynomials of different degrees fitting an artificially generated dataset (left) and the corresponding density $p(t|\mathbf{x}_n, \mathbf{w}, \sigma^2)$ in correspondence of the selected datapoint $\mathbf{x}_n$. The red line shows the selected point on the left plot while it corresponds to the value of t_n on the right one. The values where it intersects the Gaussian densities correspond to the likelihood of the three models, since the quadratic function is the closest one it has the highest likelihood.

In order to optimize the parameters over the whole data set it is necessary to maximize the joint likelihood $p(t_1, ..., t_N|\mathbf{w}, \sigma^2, \mathbf{x}_1, ..., \mathbf{x}_N)$, which can be factorized under the assumption of independence of the observations:

$$p(\mathbf{t}|\mathbf{X}, \mathbf{w}, \sigma^2) = \prod_{n=1}^{N} p(t_n|\mathbf{x}_n, \mathbf{w}, \sigma^2)$$

where $\mathbf{t} = (t_1, ..., t_N)^\top$ and $\mathbf{X}$ is the $N \times D$ matrix where each row $\mathbf{x}_n$ is the feature vector of the corresponding observation. The maximization of the likelihood is usually not trivial and requires the use of iterative algorithms such as gradient descent (Ruder, 2016). However, it is possible to find a closed form solution for linear regression. It is generally easier to work with the logarithm of the likelihood, so that the logarithm of the product can be transformed in the sum of the logarithms:

$$\underset{\mathbf{w},\sigma^2}{\text{argmax}} \log \prod_{n=1}^{N} p(t_n|\mathbf{x}_n, \mathbf{w}, \sigma^2) = \underset{\mathbf{w},\sigma^2}{\text{argmax}} \sum_{n=1}^{N} \log p(t_n|\mathbf{x}_n, \mathbf{w}, \sigma^2) =$$

$$= \underset{\mathbf{w},\sigma^2}{argmax} \sum_{n=1}^{N} \log \frac{1}{\sigma\sqrt{2\pi}} \exp\left\{-\frac{1}{2\sigma^2}(t_n - \mathbf{w}^\top \mathbf{x}_n)^2\right\} =$$

$$= \underset{\mathbf{w},\sigma^2}{argmax} \sum_{n=1}^{N} \log \frac{1}{\sigma\sqrt{2\pi}} - \sum_{n=1}^{N} \frac{1}{2\sigma^2}(t_n - \mathbf{w}^\top \mathbf{x}_n)^2 =$$

$$= \underset{\mathbf{w},\sigma^2}{argmax} - N \log \sigma\sqrt{2\pi} - \frac{1}{2\sigma^2} \sum_{n=1}^{N} (t_n - \mathbf{w}^\top \mathbf{x}_n)^2 =$$

$$= \underset{\mathbf{w},\sigma^2}{argmax} - N \log \sigma\sqrt{2\pi} - \frac{1}{2\sigma^2}(\mathbf{t} - \mathbf{X}\mathbf{w})^\top(\mathbf{t} - \mathbf{X}\mathbf{w})$$

In the case of linear regression with Gaussian noise, the likelihood function is convex, therefore it is possible to find the optimal values by setting the partial derivatives with respect to $\mathbf{w}$ and σ to zero:

$$\frac{\partial \log \mathrm{L}}{\partial \mathbf{w}} = -\frac{1}{\sigma^2}(\mathbf{X}^\top \mathbf{X}\mathbf{w} - \mathbf{X}^\top \mathbf{t}) = 0$$

$$\frac{\partial \log \mathrm{L}}{\partial \sigma^2} = -\frac{N}{\sigma^2} + \frac{1}{(\sigma^2)^2}(\mathbf{t} - \mathbf{X}\mathbf{w})^\top(\mathbf{t} - \mathbf{X}\mathbf{w}) = 0$$

$$\mathbf{w} = (\mathbf{X}^\top \mathbf{X})^{-1}\mathbf{X}^\top \mathbf{t}, \qquad \sigma^2 = \frac{1}{N}(\mathbf{t} - \mathbf{X}\mathbf{w})^\top(\mathbf{t} - \mathbf{X}\mathbf{w})$$

The optimal value of $\mathbf{w}$ is the same that we would have got by minimizing the Mean Squared Error (MSE) function, while the value of the variance corresponds to the MSE itself. This solution suggests a strong relation between MSE minimization and likelihood maximization, even if the latter models $\mathbf{t}$ as a random variable, it still has the same issues of the former, as it will be showed in Section 2.2.

2.1.2 Parameters uncertainty

In order to understand how confident we should be on the model parameters $\mathbf{w}$ and σ^2, they should be compared with the corresponding expected values because they depend on the available observations which are just a sample of the infinite possible observations we could get from the original model $p(\mathbf{t}|\mathbf{X}, \mathbf{w}, \sigma^2) = \mathscr{N}(\mathbf{X}\mathbf{w}, \sigma^2)$.

Assuming that we have the original parameters $\mathbf{w}$ and σ^2, the expectation of the maximum likelihood estimates $\hat{\mathbf{w}}$ and $\hat{\sigma}^2$ is:

$$\mathbb{E}_{p(\mathbf{t}|\Phi,\mathbf{w},\sigma^2)}\{\hat{\mathbf{w}}\} = \int \hat{\mathbf{w}} p(\mathbf{t}|\Phi, \mathbf{w}, \sigma^2) dt =$$

$$= \int (\Phi^\top \Phi)^{-1} \Phi^\top \mathbf{t} p(\mathbf{t}|\Phi, \mathbf{w}, \sigma^2) dt =$$

$$= (\Phi^\top \Phi)^{-1} \Phi^\top \mathbb{E}_{p(\mathbf{t}|\Phi,\mathbf{w},\sigma^2)}\{\mathbf{t}\} =$$

$$= (\Phi^\top \Phi)^{-1} \Phi^\top \Phi \mathbf{w} = \mathbf{w}$$

$$\mathbb{E}_{p(\mathbf{t}|\Phi,\mathbf{w},\sigma^2)}\{\hat{\sigma}^2\} = \int \hat{\sigma}^2 p(\mathbf{t}|\Phi,\mathbf{w},\sigma^2)dt =$$

$$= \int \frac{1}{N}(\mathbf{t}-\Phi\hat{\mathbf{w}})^\top(\mathbf{t}-\Phi\hat{\mathbf{w}})p(\mathbf{t}|\Phi,\mathbf{w},\sigma^2)dt =$$

$$= \sigma^2\left(1-\frac{M}{N}\right)$$

The results show that the value of $\mathbf{w}$ is unbiased while σ^2 is generally lower than the expected value, therefore the model is overconfident about its predictions. Moreover the formulation of the expected value of the variance suggests that by increasing the number of observations N, the value gets closer to the expectation, while more complex models, meaning a higher number of parameters M, need bigger amounts of data and generally increase the bias.

Figure 2.3 shows how the expected value of the variance get closer to the its real value as the model sees more data. Moreover, it shows how polynomials of higher degrees have lower variance than the lower order ones after processing the same amount of data.

Since great amounts of data are not always available to learn the best parameters, it is not possible to be certain about the variance of the data. The next section shows that, in the same way, even with a large dataset, there is still uncertainty about the model parameters as well as the model choice.

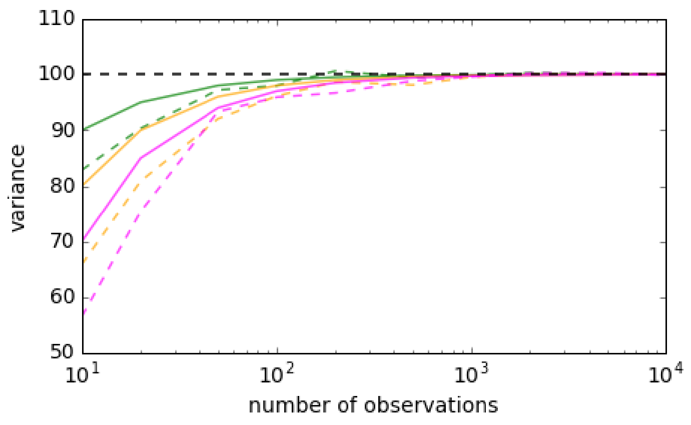


Fig. 2.3 Theoretical expected value of the variance (solid lines) and expected value computed by averaging over 100 samples (dashed lines), for different number of observations, for polynomials of 1st (green), 2nd (orange) and 3rd (magenta) degree. The dashed black line corresponds to the real value of the variance used to generate the observations.

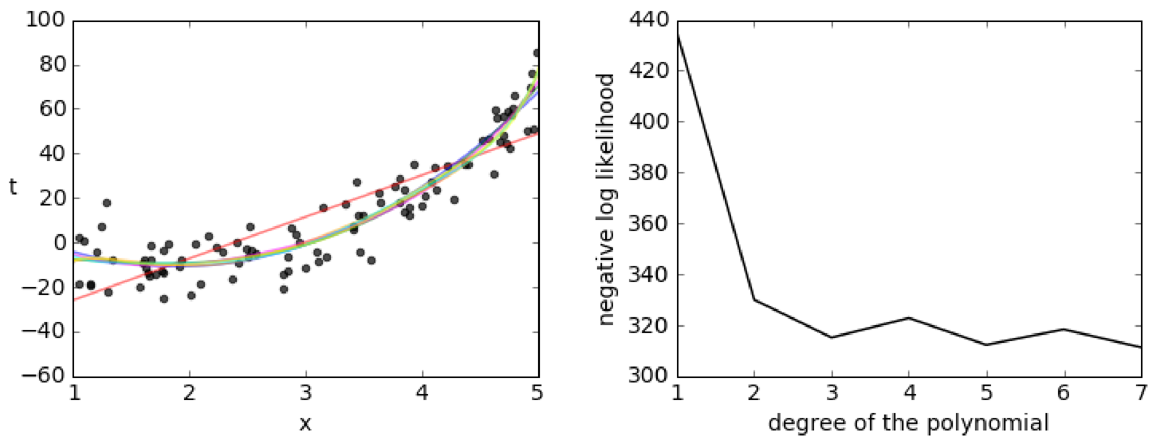


Fig. 2.4 Polynomials of different orders fitting artificially generated data (left) and the corresponding negative log likelihood (right).

2.2 Overfitting

In order to better fit the data it is possible to give the model more flexibility by transforming the input features using basis functions (Bishop, 2006), the new features are $\phi(x) = (1, x, x^2, ..., x^q)$ where q is the degree of the selected polynomial, and the problem formulation and solution are unchanged:

$$\underset{\mathbf{w},\sigma^2}{argmax} \prod_{n=1}^{N} p(t_n|\mathbf{w}, \sigma^2, \phi(x_n))$$

$$\mathbf{w} = (\Phi^\top \Phi)^{-1} \Phi^\top \mathbf{t}, \qquad \sigma^2 = \frac{1}{N}(\mathbf{t} - \Phi\mathbf{w})^\top (\mathbf{t} - \Phi\mathbf{w})$$

where Φ is the $N \times M$ *design matrix* where each row is the vector of features obtained by transforming one observation, $\Phi_n = (\phi_1(x_n), ..., \phi_M(x_n))$.

Figure 2.4 illustrates different polynomials fitting a set of data points, the negative log likelihood decreases as the polynomial's order increases, suggesting that more complex models better fit the data.

Figure 2.5 shows the same data as Figure 2.4 but this time only part of the data has been used to optimize the model parameters while the rest has been kept to measure the prediction error. It is easy to verify that more complex models only fit the training data better, in fact, as the negative log likelihood decreases with the degree of the polynomial, the test MSE increases. This is a well known problem under the name of *overfitting* (Caruana et al., 2001; Lever et al., 2016): as the model gets more complex, it also gets too specific on the training data and does not generalize well.

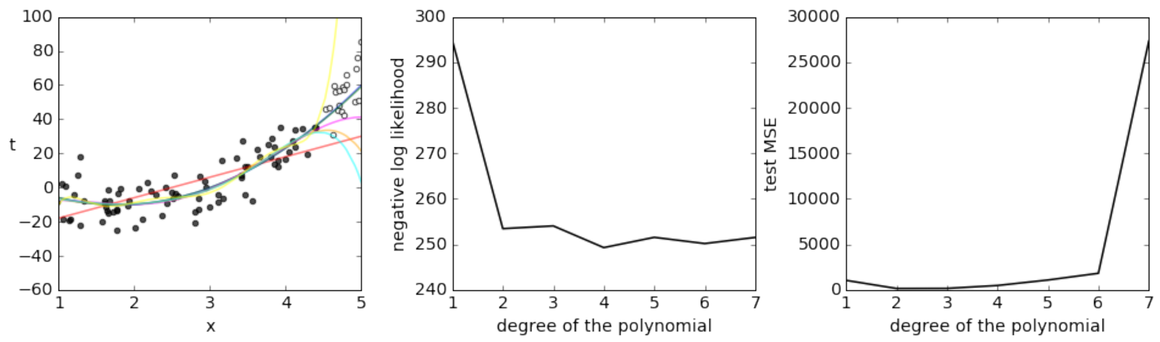


Fig. 2.5 Polynomials of different orders fitting training data in black (left) and the corresponding negative log likelihood (center) and mean squared error calculated on the test data in white (right).

2.2.1 Predictive variance

Given a new input $\mathbf{x}_{new}$, the corresponding target will be computed as follows:

$$t_{new} = \mathbf{w}^\top \phi(\mathbf{x}_{new})$$

In the same way as we computed the expectation of σ^2 and $\mathbf{w}$ we can compute the expectation over the predictions in order to understand how confident they are:

$$\mathbb{E}_{p(\mathbf{t}|\Phi,\mathbf{w},\sigma^2)}\{t_{new}\} = \mathbb{E}_{p(\mathbf{t}|\Phi,\mathbf{w},\sigma^2)}\{\hat{\mathbf{w}}^\top \phi(\mathbf{x}_{new})\} =$$

$$= \mathbb{E}_{p(\mathbf{t}|\Phi,\mathbf{w},\sigma^2)}\{((\Phi^\top\Phi)^{-1}\Phi^\top\mathbf{t})^\top\}\phi(\mathbf{x}_{new}) =$$

$$= \mathbb{E}_{p(\mathbf{t}|\Phi,\mathbf{w},\sigma^2)}\{((\Phi^\top\Phi)^{-1}\Phi^\top\Phi\mathbf{w})^\top\}\phi(\mathbf{x}_{new}) =$$

$$= \mathbb{E}_{p(\mathbf{t}|\Phi,\mathbf{w},\sigma^2)}\{\mathbf{w}^\top\}\phi(\mathbf{x}_{new}) = \mathbf{w}^\top\phi(\mathbf{x}_{new})$$

The value of $\mathbf{w}$ is unbiased and this reflects on the prediction which are unbiased as well. Moreover, since $\mathbf{t}$ has been modeled as a random variable, it is possible to estimate the variance of new predictions:

$$\mathrm{var}\{t_{new}\} = \mathbb{E}_{p(\mathbf{t}|\Phi,\mathbf{w},\sigma^2)}\{t_{new}^2\} - \mathbb{E}_{p(\mathbf{t}|\Phi,\mathbf{w},\sigma^2)}\{t_{new}\}^2 =$$

$$= \sigma^2\phi(\mathbf{x}_{new}^\top)(\Phi^\top\Phi)^{-1}\phi(\mathbf{x}_{new})$$

This final equation shows how the farther we get from the original training data, the less confidence we have in the predictions, this can be better visualized in Figure 2.6 where the coloured boundaries correspond to an interval of two standard deviations around the predicted value.

Moreover, the prediction variance can be rewritten using the covariance of the parameters:

$$\mathrm{cov}\{\hat{\mathbf{w}}\} = \sigma^2(\Phi^\top\Phi)^{-1}$$

$$\mathrm{var}\{t_{new}\} = \phi(\mathbf{x}_{new}^\top)\mathrm{cov}\{\hat{\mathbf{w}}\}\phi(\mathbf{x}_{new})$$

showing how a high uncertainty in the parameters will lead to high uncertainty in the predictions as illustrated again in Figure 2.6.

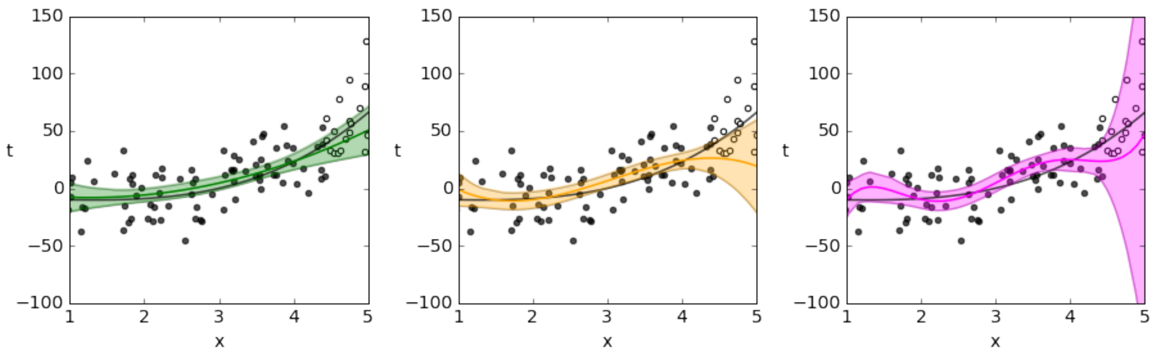


Fig. 2.6 Predictive variance for polynomials of 2nd, 3rd and 5th order respectively. The black line corresponds to the model used to generate the data.

2.2.2 Regularization and validation

Even if it is possible to make predictions and quantify their uncertainty, these measures still heavily depend on the selected model and its parameters which could be overfitting. Different solutions exist to avoid this problem which can be divided in regularization and validation techniques.

Given a complex model, regularization aims at keeping it as simple as possible by constraining the values of its parameters, in order to do so, a regularization term is added to the original loss function.

$$\hat{\mathbf{w}} = \underset{\mathbf{w}}{argmin}\ \mathrm{Loss}(\mathbf{X},\mathbf{w}) + \lambda \cdot \mathrm{Regularization}(\mathbf{w})$$

where λ is an hyperparameter used to weight the regularization term. The most common regularization term is the sum of the squared values of the parameters: $\mathbf{w}^\top\mathbf{w}$, it is called *weight decay* (or L2) because it encourages lower values (Krogh and Hertz, 1992). Another

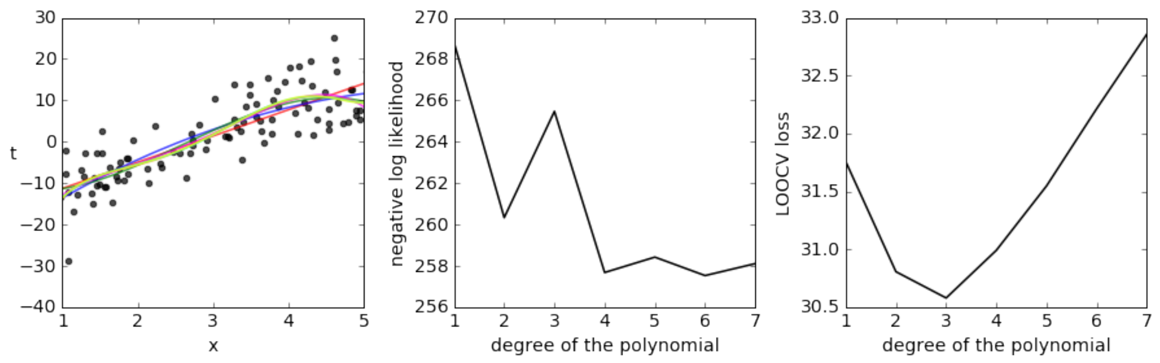


Fig. 2.7 Polynomials of different orders fitting artificially generated data (left) and corresponding negative log likelihood (center) and LOOCV error (right).

common regularization is the *lasso* (or L1): $|\mathbf{w}|$ which instead leads to sparsity in the model parameters (Tibshirani, 1996). In order to make regularization techniques effective it is necessary to find a good value for the hyperparameter λ. If it is too small, then the regularization has no effect, while if it is too big, the model is too constrained and it is not able to capture the real source of the data.

Validation can be used both for model selection and hyperparameter tuning in order to avoid overfitting. In its simplest form it evaluates the model performance on a portion of data that has not been used for the training and will not be used at test time, its only aim is to validate intermediate model performance (Kohavi et al., 1995).

In order to decrease the bias due to the particular choice of the validation set, *k*-fold cross validation can be used. It splits the data set in *k* equal folds and iteratively trains the model on *k*-1 folds and measures the error on the last one. Finally, the mean of the error on the different folds can be used for model comparison.

A finer decomposition of the data leads to a more accurate validation. Leave-One-Out Cross Validation (LOOCV) validates the model keeping out just one observation per time. (Wong, 2015) An example of this technique is showed in Figure 2.7, while the negative log likelihood decreases as the model becomes more complex, the LOOCV is able to select the right degree of the polynomial.

The main drawback of k-fold CV and LOOCV is the need to train the model multiple times which can be very computationally expensive, leading to the use of a lower number of folds.

2.3 Bayesian Linear Regression

Standard linear regression models are able to quantify confidence in the predictions but they only use a point estimate of the model parameters. Model selection approaches are very expensive and require to sacrifice part of the initial data while it is still possible that the optimized model is overfitting.

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 final predictions are made by weighting the predictions made with every parameters setting using their corresponding probability.

In the case of linear regression we could assume that the distribution of the parameters is Gaussian as illustrated in Figure2.8 (left). Each sample from this distribution corresponds to the set of parameters of a model that can be used to make predictions. The prediction of the yellow line if Figure 2.8 (right) will have higher weight that the green and magenta ones because the corresponding model parameters have higher probability.

2.3.1 Bayes theorem

The distribution of the model parameters could be defined using the mean and covariance of **w** as previously computed but, as already seen in the previous sections, this is prone to

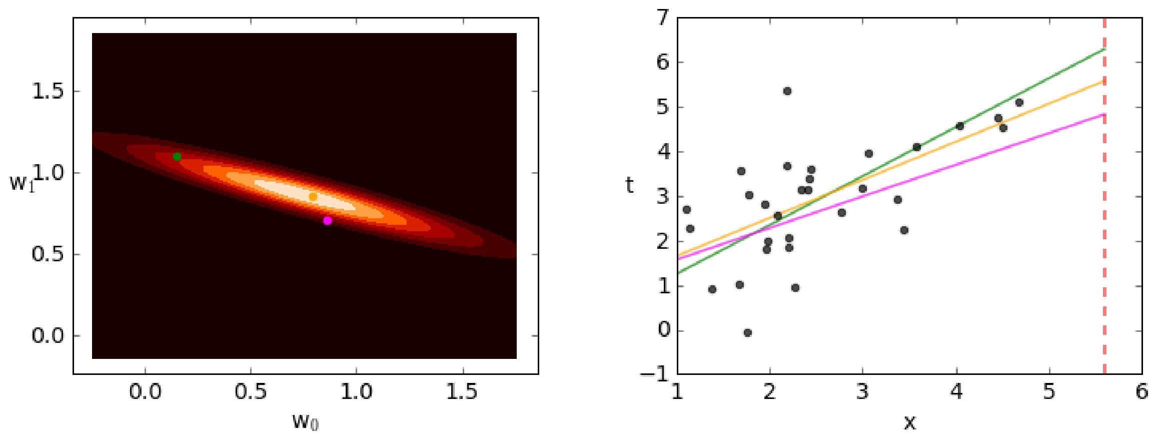


Fig. 2.8 Density of the model parameters (left) and prediction of the value on the red dahsed line using three models with different parameters (right). Each model corresponds to a sample from the prameters distribution.

overfitting. Another possibility is to use Bayes theorem:

$$p(A|B) = \frac{p(B|A)p(A)}{p(B)}$$

$$p(\mathbf{w}|\mathbf{X}, \mathbf{t}, \sigma^2) = \frac{p(\mathbf{t}|\mathbf{X}, \mathbf{w}, \sigma^2)p(\mathbf{w})}{p(\mathbf{t}|\mathbf{X}, \sigma^2)}$$

This distribution is called *posterior* distribution, that is the distribution over the model parameters, conditioned on the training data. Its computation requires the likelihood $p(\mathbf{t}|\mathbf{X}, \mathbf{w}, \sigma^2)$, already used before, and the *prior* density $p(\mathbf{w})$. The latter encodes one of the key advantages of the use of the Bayesian approach: it allows to use previous knowledge about the parameters. This distribution should therefore express the prior beliefs we have on the parameters.

Finally, the marginal likelihood $p(\mathbf{t}|\mathbf{X}, \sigma^2)$ is the normalization constant used to ensure that the posterior respects the properties of a probability distribution. It is computed has follows:

$$p(\mathbf{t}|\mathbf{X}, \sigma^2) = \int p(\mathbf{t}|\mathbf{X}, \mathbf{w}, \sigma^2)p(\mathbf{w})d\mathbf{w}$$

therefore it can be interpreted as a likelihood measure which is not fixed on the choice of the parameters $\mathbf{w}$ because it considers them all. For this reason, it can be used for model comparison without the need for further validation.

2.3.2 Conjugate prior and likelihood

Solving the marginal likelihood integral is also known as the marginalization (or integration) problem. It complicates the posterior computation because it is often either intractable or very computationally expensive (Bos, 2002) as it will be discussed in Section 2.4.

However, for simpler models like linear regression, it is possible to choose a conjugate prior for the likelihood, so that, by definition of conjugacy, the posterior distribution is in the same family of the prior and the marginalization problem can be solved analytically.

There exist different examples of conjugate distributions, both for the discrete case, e.g. Beta prior for Bernoulli likelihood, and the continuous one, e.g. Gamma prior and likelihood. We already modeled the noise in the targets as a normal distribution, resulting in a Gaussian likelihood for the linear regression model, hence a possible choice for the conjugate prior is a Gaussian distribution as well. As already said, the prior is used to encode previous knowledge, therefore the values of the prior hyperparameters are design choices. For example, if we were to use a Beta prior distribution for the problem of modeling the distribution of heads and tails in a coin tossing experiment, then the prior hyperparameters α

and β can be used to encode some pseudo-observation corresponding to $\alpha-1$ heads and $\beta-1$ tails.

Since for the linear regression problem we have no particular previous knowledge, it is good practice to use a wide prior, so that it doesn't favor any particular value. A better discussion of the prior choice is presented in Section 3.3.2

In order to compute the posterior mean and variance, we first need to look at the shape of the posterior, excluding the terms that are not a function of the model parameters:

$$p(\mathbf{w}|\mathbf{X},\mathbf{t},\sigma^2) = \mathcal{N}(\mu,\Sigma)$$

$$p(\mathbf{w}|\mathbf{X},\mathbf{t},\sigma^2) \propto \exp\left\{-\frac{1}{2}(\mathbf{w}-\mu)^\top\Sigma^{-1}(\mathbf{w}-\mu)\right\} \propto \exp\left\{-\frac{1}{2}(\mathbf{w}^\top\Sigma^{-1}\mathbf{w}-2\mathbf{w}^\top\Sigma^{-1}\mu)\right\}$$

Following the same approach, we can rewrite the posterior given the prior and the likelihood, and complete the square in the exponential ignoring the normalizing constants:

$$p(\mathbf{w}|\mathbf{X},\mathbf{t},\sigma^2) = \frac{p(\mathbf{t}|\mathbf{X},\mathbf{w},\sigma^2)p(\mathbf{w})}{p(\mathbf{t}|\mathbf{X},\sigma^2)}$$

$$p(\mathbf{t}|\mathbf{X},\mathbf{w},\sigma^2) = \mathcal{N}(\mathbf{X}\mathbf{w},\sigma^2\mathbf{I}), \qquad p(\mathbf{w}) = \mathcal{N}(\mu_0,\mathbf{S}_0)$$

$$p(\mathbf{w}|\mathbf{X},\mathbf{t},\sigma^2) \propto \exp\left\{-\frac{1}{2\sigma^2}(\mathbf{t}-\mathbf{X}\mathbf{w})^\top(\mathbf{t}-\mathbf{X}\mathbf{w})\right\}\exp\left\{-\frac{1}{2}(\mathbf{w}-\mu_0)^\top\mathbf{S}_0^{-1}(\mathbf{w}-\mu_0)\right\} \propto$$

$$\propto \exp\left\{-\frac{1}{2}\left[\mathbf{w}^\top\left(\frac{1}{\sigma^2}\mathbf{X}^\top\mathbf{X}+\mathbf{S}_0^{-1}\right)\mathbf{w}+\mathbf{w}^\top\left(\frac{1}{\sigma^2}\mathbf{X}^\top\mathbf{t}+\mathbf{S}_0^{-1}\mu_0\right)\right]\right\}$$

By matching the two expressions of the posterior exponential we can find the posterior parameters:

$$\Sigma^{-1} = \frac{1}{\sigma^2}\mathbf{X}^\top\mathbf{X}+\mathbf{S}_0^{-1}, \qquad \mu = \Sigma\left(\frac{1}{\sigma^2}\mathbf{X}^\top\mathbf{t}+\mathbf{S}_0^{-1}\mu_0\right)$$

From this solution it is possible to notice that, when the prior is infinitely broad, meaning $\mathbf{S}_0^{-1}\to 0$, the mean of the posterior corresponds to the MLE solution $\mathbf{w} = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{t}$ because no previous knowledge and beliefs have been applied.

Since we will consider a fairly wide prior and we have no previous knowledge about the parameters, the prior distribution will be an isotropic Gaussian with zero mean:

$$p(\mathbf{w}|\alpha) = \mathcal{N}(\mathbf{0},\alpha^{-1}\mathbf{I})$$

where α is the precision, that is the inverse of the variance. By using this prior distribution and setting $\beta = (\sigma^2)^{-1}$, we can finally write the posterior parameters as follows:

$$\Sigma^{-1} = \beta \mathbf{X}^\top \mathbf{X} + \alpha \mathbf{I}, \qquad \mu = \beta \Sigma \mathbf{X}^\top \mathbf{t}$$

2.3.3 Predictive distribution

The posterior distribution can provide useful insights about the model parameters but its main use is to compute the predictive distribution. As introduced at the beginning of this section, in order to make predictions we consider the predictions made using all the possible parameters settings, weighted by their probability which is given by the posterior:

$$p(t_{new}|x_{new}, \mathbf{X}, \mathbf{t}, \beta, \alpha) = \int p(t_{new}|x_{new}, \mathbf{w}, \beta) p(\mathbf{w}|\mathbf{X}, \mathbf{t}, \beta, \alpha) d\mathbf{w}$$

Since both the terms in the integral are normal distributions, $p(\mathbf{w}|\mathbf{X}, \mathbf{t}, \beta, \alpha) = \mathcal{N}(\mu, \Sigma)$ and $p(t_{new}|x_{new}, \mathbf{w}, \beta) = \mathcal{N}(\mathbf{w}^\top x_{new}, \beta^{-1})$, this integral can be solved analytically and the predictive distribution is a Gaussian as well (Bishop, 2006):

$$p(t_{new}|x_{new}, \mathbf{X}, \mathbf{t}, \beta, \alpha) = \mathcal{N}(\mu^\top x_{new}, \beta^{-1} + x_{new}^\top \Sigma^{-1} x_{new})$$

The predictive variance is the sum of two independent components, the first one is associated to the noise in the data (β^{-1}) and, for this reason, it is constant and its value is more accurate as we get more training data, as discussed in Section 2.1.2. The second component depends on the covariance of the parameters, hence an uncertain model will lead to uncertain predictions. Moreover, as the number of observations N increases, the uncertainty in the parameters decreases and the variance of the predictive distribution converges to β^{-1}.

Figure 2.9 shows the evolution of the posterior distribution and the corresponding model as the number of observation increases. On the first column we can see the posterior getting narrower and centering on the value corresponding to the original model. The second column illustrates how, as the model sees more observations, the prior looses importance and the samples get less general and closer to the original model. Finally, the last column displays the decrease of the predictive variance and how it gets larger as the it get farther from the original observations.

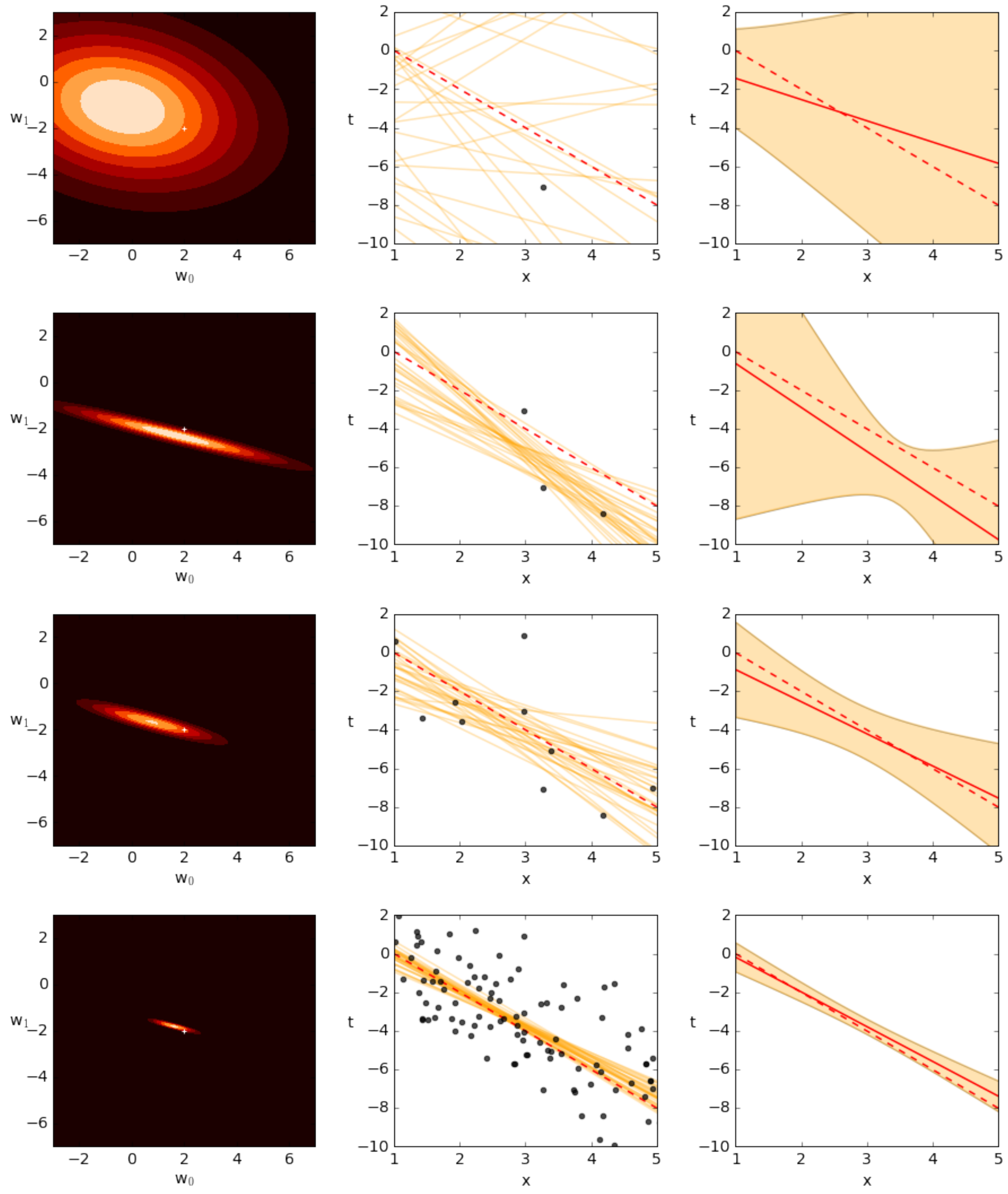


Fig. 2.9 The first column illustrates the posterior distribution and the original model (white cross) as the model sees more data points. The second column shows the original model (dashed line), 30 models obtained by sampling parameters from the posterior in the first column, and the data points used to compute the posterior, respectively 1, 3, 10 and 100. The third column displays the Bayesian model with uncertainty boundaries corresponding to 2 standard deviations, and the original model.

2.4 Bayesian Neural Networks

To introduce Bayesian linear regression we started by presenting linear regression, therefore in order to explain Bayesian neural networks we must first analyze the relation between linear regression and neural networks. In particular, we will consider linear basis functions regression because it is more flexible and closer to the neural network case. As already discussed in Section 2.2, linear basis functions regression is equivalent to performing linear regression on a new set of features obtained by applying basis functions to the initial features vector: $\boldsymbol{\phi}(\mathbf{x}) = (\phi_1(\mathbf{x}), ..., \phi_M(\mathbf{x}))$. Moreover, it is possible to parameterize these basis function so that $\phi_m(\mathbf{x}) = f(\mathbf{w}_m^\top \mathbf{x} + b_m)$, in this case the basis functions f are all the same for every m, e.g. $\phi_m(\cdot) = sin(\cdot)$ or $\phi_m(\cdot) = exp(\cdot)$.

We can define a neural network as a stack of linear basis functions regressions where each new set of features is called *layer*, the basis functions are called *activation* functions and the parameters $\mathbf{w}$ and b are the weights and biases of each layer. This model can be used both for regression, as just seen, and classification, by composing a logistic function on the final layer.

In order to apply the Bayesian approach to this model we first need to define a likelihood and a prior distribution. Because of the activation functions which are not linear, the likelihood of the model is not Gaussian anymore, and there are no possible conjugate priors. The main issue of Bayesian modeling for complex models such as neural networks is the previously defined marginalization problem: the marginal likelihood cannot be computed hence the posterior distribution cannot be obtained.

The posterior distribution shape is unknown but it is still possible to compute the value of $p(\mathbf{t}|\mathbf{X}, \mathbf{w}, \beta, \alpha)p(\mathbf{w}|\alpha)$ which is proportional to the value of the posterior. Maximizing this product would give the value that maximizes the posterior as well, also known as maximum a posteriori (MAP) solution. This value is still just a point estimate of the parameters instead of a distribution, therefore the uncertainty is not modeled.

2.4.1 The Laplace approximation

A possible solution to the posterior computation problem is to try to approximate it with another distribution. The Laplace approximation tries to fit a Gaussian distribution on the posterior (Azevedo-Filho and Shachter, 1994; Laplace, 1986; MacKay, 1992). In order to do so, it assumes that the posterior distribution has the following shape:

$$p(z) = \frac{1}{Z} f(z), \qquad Z = \int f(z) dz$$

then it places the mean of the approximating Gaussian in the same place as one of the posterior's modes, which corresponds to the MAP value.

Since the logarithm of a Gaussian distribution is a quadratic function of the variables, it can be reformulated using the Taylor expansion:

$$lnf(z) \simeq lnf(z_0) - \frac{1}{2}A(z-z_0)^2, \qquad A = -\left.\frac{d^2}{dz^2}lnf(z)\right|_{z=z_0}$$

where the first order term is not present because z_0 is a local maximum of the distribution. By taking the exponential and normalizing using the standard Gaussian normalization, we obtain the Laplace approximation:

$$f(z) \simeq f(z_0)\,\exp\left\{-\frac{A}{2}(z-z_0)^2\right\}$$

$$q(z) = \left(\frac{A}{2\pi}\right)^{\frac{1}{2}} \exp\left\{-\frac{A}{2}(z-z_0)^2\right\} = \mathcal{N}(z_0, A^{-1})$$

This result can be easily extended to the multivariate case, the approximating distribution is:

$$q(\mathbf{w}) = \frac{|\mathbf{A}|^{\frac{1}{2}}}{(2\pi)^{\frac{M}{2}}}\exp\left\{-\frac{1}{2}(\mathbf{z}-\mathbf{z}_0)^\top \mathbf{A}(\mathbf{z}-\mathbf{z}_0)\right\} = \mathcal{N}(\mathbf{z}_0, \mathbf{A}^{-1})$$

where the precision $\mathbf{A}$ is the Hessian matrix and needs to be positive definite in the same way as A should be positive, because $\mathbf{z}_0$ has to be a local maximum.

The main issue of this approximation is it is only able to fit one mode of the posterior and it also fits badly very skewed distributions as illustrated in Figure 2.10. Since predictions are computed using not only the mean of the posterior but also other samples, this approximation could lead to assign high probabilities to very bad parameters, therefore corrupting the quality of the predictions.

A more flexible approach to approximate the posterior distribution is to use variational inference. It allows to specify any approximating distribution $q(z)$ and then tries to fit the posterior following an iterative optimization procedure (Blei et al., 2017). Variational inference will be discussed in more detail in the next chapter.

2.4.2 Markov Chain Monte Carlo methods

Instead of approximating the posterior, an alternative solution is to sample from it using Markov chain Monte Carlo methods(Neal, 1993). Successively, since this does not provide

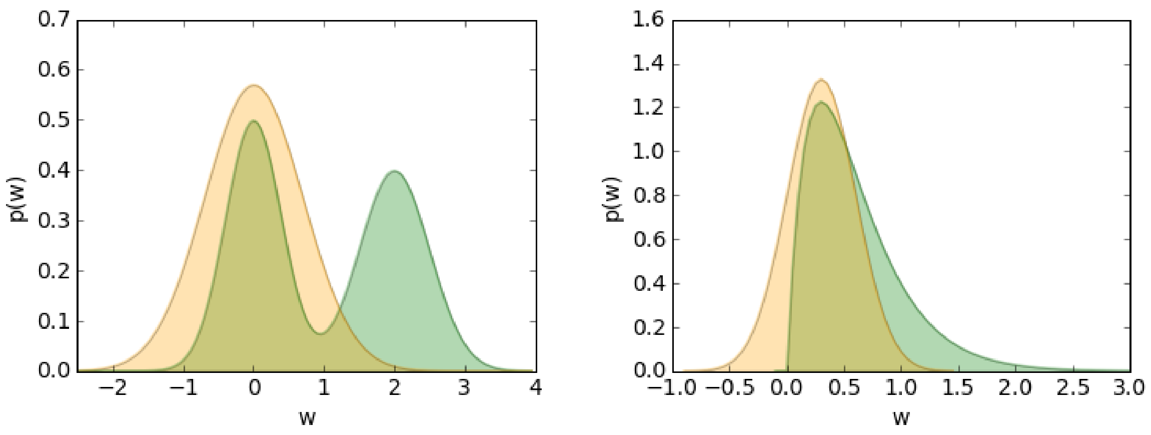


Fig. 2.10 Laplace approximation for a bimodal distribution (left) and a gamma distribution (right).

the posterior distribution, the predictive distribution needs to be approximated using only samples:

$$p(t_{new}|x_{new},\mathbf{X},\mathbf{t},\beta,\alpha) = \int p(t_{new}|x_{new},\mathbf{w},\beta)p(\mathbf{w}|\mathbf{X},\mathbf{t},\beta,\alpha)d\mathbf{w}$$

$$p(t_{new}|x_{new},\mathbf{X},\mathbf{t},\beta,\alpha) \approx \sum_{s=1}^{S} p(t_{new}|x_{new},\mathbf{w}_s,\beta), \qquad \mathbf{w}_s \sim p(\mathbf{w}|\mathbf{X},\mathbf{t},\beta,\alpha)$$

MCMC methods use a Markov chain that has the desired distribution as equilibrium distribution, so that the more samples are taken, the better the distribution is approximated. An example is the Metropolis-Hastings algorithm (Hastings, 1970) which works in two phases: given an initial sample $\mathbf{w}_s$, it first makes a proposal for the next sample $\tilde{\mathbf{w}}$ by sampling from the neighbourhood of the initial sample. Then, it decides if the new sample is to be accepted, $\mathbf{w}_{s+1} = \tilde{\mathbf{w}}$, or to be discarded, $\mathbf{w}_{s+1} = \mathbf{w}_s$, based on a given criterion and it starts again.

The usual neighbourhood distribution used in the proposal phase is a Gaussian centered on $\mathbf{w}_s$:

$$p(\tilde{\mathbf{w}}|\mathbf{w}_s,\Sigma_N) = \mathcal{N}(\mathbf{w}_s,\Sigma_N)$$

where the choice of the covariance matrix Σ_N determines how far on average can each step of the algorithm arrive. If it is too wide, the algorithm will take more time taking enough samples because of high rejection rate. If it is too low, it will take too much time to explore the distribution space or get stuck on a local mode.

The acceptance criterion for new samples is based on the following ratio:

$$r = \frac{p(\tilde{\mathbf{w}} \rightarrow \mathbf{w}_s)}{p(\mathbf{w}_s \rightarrow \tilde{\mathbf{w}})} = \frac{p(\tilde{\mathbf{w}}|\mathbf{X},\mathbf{t},\beta,\alpha)p(\mathbf{w}_s|\tilde{\mathbf{w}},\Sigma_N)}{p(\mathbf{w}_s|\mathbf{X},\mathbf{t},\beta,\alpha)p(\tilde{\mathbf{w}}|\mathbf{w}_s,\Sigma_N)}$$

which requires to evaluate the posterior for both $\mathbf{w}_s$ and $\tilde{\mathbf{w}}$. The problem of evaluating the posterior is the computation of the marginal likelihood but, since it appears both at the numerator and denominator of the fraction, it can be simplified, and the ratio can be defined as follows:

$$r = \frac{p(\mathbf{t}|\mathbf{X},\tilde{\mathbf{w}},\beta,\alpha)p(\tilde{\mathbf{w}}|\alpha)p(\mathbf{w}_s|\tilde{\mathbf{w}},\Sigma_N)}{p(\mathbf{t}|\mathbf{X},\mathbf{w}_s,\beta,\alpha)p(\mathbf{w}_s|\alpha)p(\tilde{\mathbf{w}}|\mathbf{w}_s,\Sigma_N)}$$

Finally, the new sample will be accepted with probability $p = min(1,r)$ and rejected with probability $1-p$. Since $r \geq 1$ when the new sample is better than the previous one, good samples will always be accepted, ensuring the convergence to one mode. On the other side, bad samples are not always discarded, so that it is possible to explore the solution space to find other modes and describe the distribution.

The Metropolis-Hastings algorithm has some complications due to the proper choice of the hyperparameters α and Σ_N. Moreover, in very high dimensional spaces, which correspond to standard use cases when using neural networks, it gets slower because it explores inefficiently (Chib and Greenberg, 1995).

A possible solution to this problem is to use the Hamiltonian Monte Carlo algorithm which uses concepts from Hamiltonian mechanics to reduce the correlation between successive samples (Neal et al., 2011). Higher distances between successive samples means a better exploration of the distribution space hence more representative samples and a lower rejection rate. Unfortunately, also HMC has issues exploring multimodal distributions with isolated

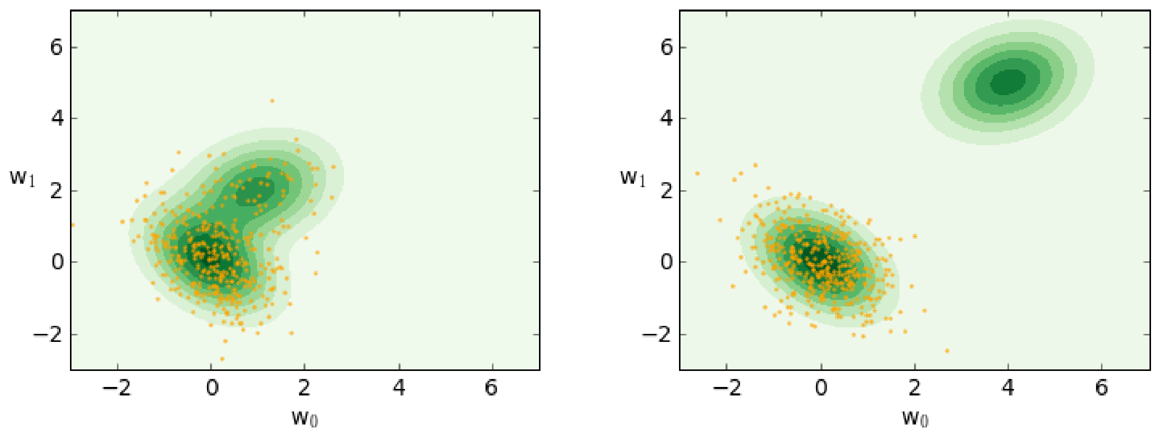


Fig. 2.11 Multimodal distribution with close modes (left) and isolated modes (right) and samples obtained using MCMC methods.

modes as illustrated in Figure 2.11 and, even if possible solutions exist to try to solve this problem, they require the use of further hyperparameters that then need to be tuned appropriately (Beskos et al., 2013; Betancourt, 2017; Girolami and Calderhead, 2011).

2.4.3 Gaussian Processes

Before concluding the section on Bayesian neural network it is worth to mention another probabilistic model because it is able to provide confidence measures and it is connected with both linear regression and Bayesian neural networks. Linear regression models can use a set of basis function to increase their flexibility but they are anyway constrained to the choice of the functions. Gaussian processes (Rasmussen, 2004) use the Bayesian approach to define a distribution over an infinite set of basis function. In order to do so, they use a kernel to move from the functions space to the observations space. This kernel is selected so that it corresponds to a space of infinite basis functions meaning that it is equivalent to performing Bayesian linear regression with an infinite set of basis functions (Gal, 2016). For the same reason a Gaussian Process can be obtained from a Bayesian Neural Network with infinite weights (Neal, 2012), and, even if this is not possible, they can still provide a good approximation while keeping the advantage of modeling the predictive uncertainty.

2.5 Advantages of Bayesian modeling

The emphasis of this work is on the importance and the advantages of uncertainty modeling in neural networks. Bayesian modeling offers a good framework to get this confidence measure because it is able to fit the data and introduce a number of useful properties while keeping the model underlying structure.

Bayesian models use a distribution over the model parameters that is coherent with the given data in fact the confidence in the parameters increases with the number of observations.

When lacking large amounts of data, the Bayesian approach is still able to model the problem appropriately by encoding prior knowledge and beliefs by means of the prior distribution. Furthermore, this distribution allows also to regularize the model parameters, as will be better proved in the next chapter.

These models are still able to fit the variance of the data and keep the dependencies on the number of available observations and the complexity of the model but the regularization is fully integrated in the problem formulation. Moreover, it is possible to perform model selection without considering a particular set of parameters but only the data because the distribution of all the possible model parameters is evaluated.

In the same way, the predictive distribution is built by weighting all the possible predictions of the given model and the predictive uncertainty increases with the "distance" from the training data.

The previous sections illustrated how this properties very often come with a cost and different trade-offs, especially for more complex models, therefore it is fundamental to evaluate the need for uncertainty estimates as well as the most appropriate implementation of the Bayesian model.

Chapter 3

Variational Inference for Bayesian Neural Networks

MCMC algorithms require great amounts of computational time to represent high dimensional distributions, and fine tuning not to get stuck on local modes of the posterior. The current trend to get uncertainty estimates is Monte Carlo dropout because of its simplicity and speed (Gal and Ghahramani, 2016). The dropout approach grounds its theoretical fundations in variational approximations for Bayesian neural networks (Gal, 2016), it is able to provide an output distribution but it does not really model the distribution of the weights (posterior distribution) that is the heart of the Bayesian approach.

For these reasons, variational inference can be a valid alternative to implement Bayesian neural networks. In this chapter we show the mathematical formulation of the Evidence Lower Bound (ELBO), which is the core of variational inference, and analyze its interpretation. Furthermore, we analyze how to extend this technique to deep neural networks and we illustrate the different challenges and trade-offs that arise in doing so.

Variational inference comes with its own issues and limitations as will be discussed in this chapter, but it also offers a good compromise between the simplicity of MC dropout and the complexity of MCMC methods, allowing to develop a well performing framework for probabilistic deep learning.

3.1 Evidence Lower Bound

As previously discussed, variational inference is one of the possible solutions to the marginalization problem to find the posterior distribution given the input data. The main idea of this approach is to approximate this distribution with a simpler one (Jordan et al., 1999). In order to do so, it is necessary to minimize the Kullback–Leibler divergence (Kullback and Leibler, 1951) between the proposed distribution and the posterior. The KL divergence is defined as follows:

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

where q is the proposed distribution, $\mathbf{w}$ is the set of model parameters, θ is the set of variational parameters and $p(\mathbf{w}|\mathscr{D})$ is the posterior distribution ($\mathscr{D}$ is now used to reference the data in a general way, without explicitly specifying $\mathbf{X}$ or $\mathbf{t}$). The variational parameters are the parameters describing the distribution of $\mathbf{w}$, in order to keep the notation clear, they will be omitted and we will refer to the posterior approximation simply as $q(\mathbf{w})$ or q.

3.1.1 Kullback–Leibler divergence

It is important to underline that the KL divergence has the following properties:

$$KL\{q(z)||p(z)\} \geq 0, \quad \forall z$$

$$KL\{q(z)||p(z)\} = 0 \quad \Leftrightarrow \quad q(z) = p(z)$$

$$KL\{q(z)||p(z)\} \neq KL\{p(z)||q(z)\}$$

Since it is asymmetric it is not a distance and minimizing the two alternative orders leads to different results as displayed in Figure 3.1. The left plot shows the *reverse* (exclusive) case that optimizes $KL\{q(z)||p(z)\}$ while the right plot shows the *forward* (inclusive) case minimizing the divergence in the opposite order. From the figure we see how in the first case the posterior approximation fits one of the modes of the distribution therefore it will not be possible to get samples from other regions of the posterior where the probability is not zero. In the second case, it tries to cover the full support of the posterior, meaning that it could get many samples from regions not belonging to the real support and it may have low probability in correspondence of the modes.

3.1.2 ELBO definition

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:

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

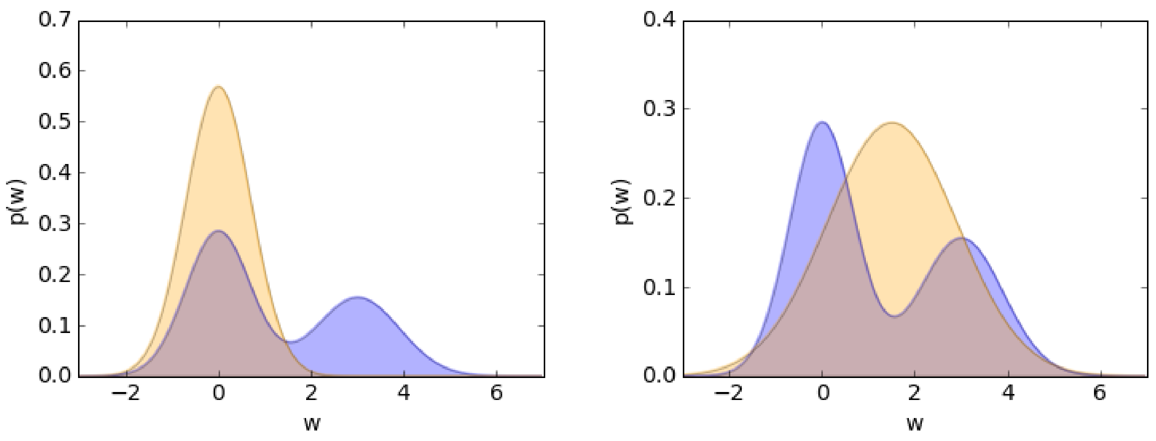


Fig. 3.1 Normal distribution (orange) minimizing the reverse (left) and the forward (right) KL divergence with a multimodal distribution (blue).

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 $q(\mathbf{w})$:

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

Then it is needed to bring the logarithm inside the expectation which can be done by applying the Jensen's inequality (Jensen, 1906):

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

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:

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

$$= \int q(\mathbf{w}) \log p(\mathscr{D}|\mathbf{w}) d\mathbf{w} - \int q(\mathbf{w}) \log \frac{p(\mathbf{w})}{q(\mathbf{w})} d\mathbf{w} =$$

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

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 $p(\mathbf{w}|\mathscr{D})$ (Jordan et al., 1999):

$$p(\mathbf{w}|\mathscr{D}) = \frac{p(\mathscr{D}|\mathbf{w})p(\mathbf{w})}{p(\mathscr{D})}, \qquad p(\mathscr{D}) = \int p(\mathscr{D}|\mathbf{w})p(\mathbf{w})d\mathbf{w}$$

$$p(t^*|x^*,\mathscr{D}) = \int p(t^*|x^*,\mathbf{w})p(\mathbf{w}|\mathscr{D})d\mathbf{w} \approx \frac{1}{S}\sum_{s=1}^{S} p(t^*|x^*,\mathbf{w}_s), \quad \mathbf{w}_s \sim p(\mathbf{w}|\mathscr{D})$$

$$KL\{q(\mathbf{w})||p(\mathbf{w}|\mathscr{D})\} = \int q(\mathbf{w})\log\frac{q(\mathbf{w})}{p(\mathbf{w}|\mathscr{D})}d\mathbf{w} = \int q(\mathbf{w})\log\frac{q(\mathbf{w})p(\mathscr{D})}{p(\mathscr{D}|\mathbf{w})p(\mathbf{w})}d\mathbf{w} =$$

$$= -\int q(\mathbf{w})\log p(\mathscr{D}|\mathbf{w})d\mathbf{w} + \int q(\mathbf{w})\log\frac{q(\mathbf{w})}{p(\mathbf{w})}d\mathbf{w} + \int q(\mathbf{w})\log p(\mathscr{D})d\mathbf{w} =$$

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

$$KL\{q(\mathbf{w})||p(\mathbf{w}|\mathscr{D})\} = -\mathscr{L}(\theta) + \log p(\mathscr{D})$$

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

3.2 Variational Inference

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). A more visual explanation of the optimization process is provided by Figure 3.2.

The area surrounded by the dashed line corresponds to the space of the variational parameters θ, each point in this space corresponds to a possible shape of the posterior approximation $q(\mathbf{w};\theta)$. For example, if the approximation is a Gaussian distribution $\mathscr{N}(\mu,\sigma)$, then this area corresponds to the space of all the possible means and standard deviations, $\theta = \{\mu,\sigma\}$. The arrow corresponds to the optimization process leading to the best variational parameters $\theta^\star$ minimizing the KL divergence from the posterior.

The choice to place the posterior distribution outside the dashed area is not casual. In our example the proposed approximation is a normal distribution therefore it cannot perfectly fit a multimodal posterior and the KL divergence will always be non-zero. This is almost always the case because it is the result of the choice of q which is a tradeoff between a distribution

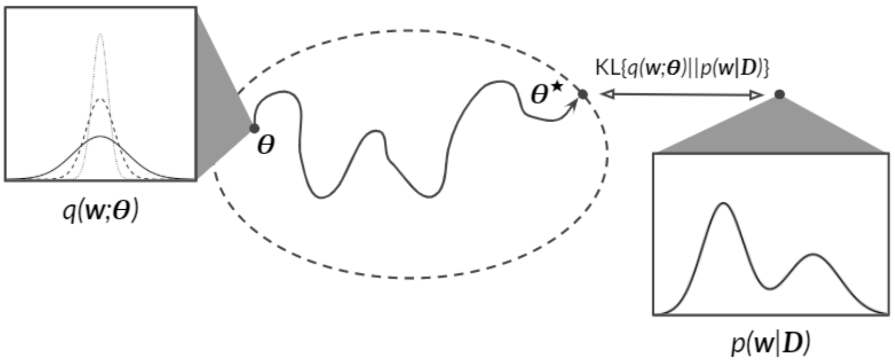


Fig. 3.2 Illustration of the minimization of the KL divergence between a multimodal posterior and its Gaussian approximation by minimizing the evidence lower bound as a function of the variational parameters.

that is able to capture the posterior shape in the best possible way, but also it needs to remain simple enough to be easy to optimize.

3.2.1 Mean field variational inference

For the reasons just listed a common choice is to use a factorized distribution, called mean field distribution, to approximate the posterior (Anderson and Peterson, 1987; Opper and Winther, 1997). This approximation assumes that all the latent variables, namely the variational parameters, are independent and therefore simplifies heavily the computation. For example, let's suppose we need to perform variational inference on the posterior of a model with M parameters and the approximating distribution is a multivariate Gaussian. Applying the mean field approximation means that the only latent variables are the mean and the variance of each parameter and the approximating distribution can be reformulated as follows:

$$p(\mathbf{w}|\mathscr{D}) \approx q(\mathbf{w}) = \prod_{i=1}^{M} q(w_i) = \prod_{i=1}^{M} N(\mu_i, \sigma_i)$$

The number of latent variables to optimize is two times the number of parameters of the original model. If we were to model the variational parameters as if they were not independent, then we would need to consider also the covariance of the joint distribution of the parameters:

$$p(\mathbf{w}|\mathscr{D}) \approx q(\mathbf{w}) = N(\mu, \Sigma)$$

Therefore the number of latent variables to optimize would be $2M + M(M-1)/2 = M(M+3)/2$ which corresponds to an enormous increase of computation that can easily become prohibitive

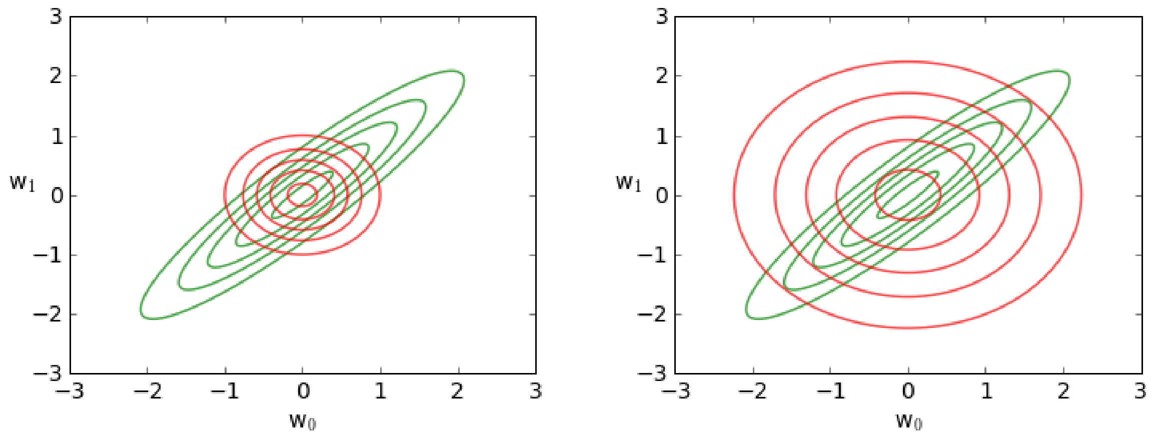


Fig. 3.3 Gaussian mean field approximations (red) of a normal distribution with high covariance (green).

for models with many parameters such as neural networks. The computational gain obtained using mean field variational inference comes at the cost of a farther approximation as illustrated in Figure 3.3 where the approximating distribution is not able to fit a bivariate Gaussian. The approximation usually fits the posterior as displayed in the left plot, hence leaving a big portion of the support uncovered.

This limitation may cause major issues in complex models where at least some portions of the parameters are correlated (Wainwright et al., 2008). Neural networks architectures have a stacked structure where the results computed using one layer's parameters are fed to the next one, hence they are very exposed to this limitation but the training process is still able to converge to a good posterior approximation, leading to almost optimal performances.

3.2.2 Stochastic variational inference

Big datasets allow to use more complex models but also introduce new scalability issues. Stochastic variational inference exploits the structure of the variational objective, i.e. the 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, 2017) and therefore is suited also for a minibatch approach (Blundell et al., 2015; Graves, 2011). The standard approach to optimize the weights of a neural network is the backpropagation algorithm (Rumelhart et al., 1986). In order to be able to apply it to the 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 reparameterization

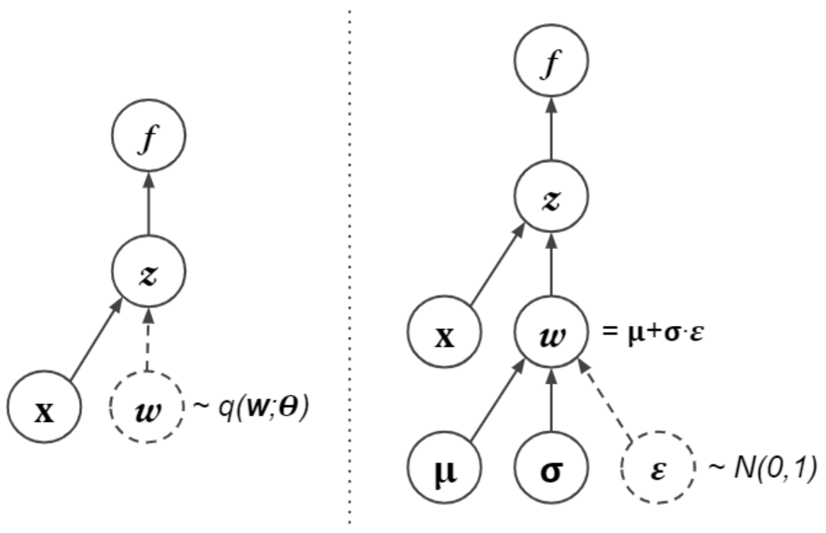


Fig. 3.4 Bayesian model before (left) and after (right) the reparameterization trick.

trick (Blundell et al., 2015; Kingma and Welling, 2013; Rezende et al., 2014) because it allows to write the weights of the neural network as a function of the variational parameters.

Following the previous example of a Gaussian posterior approximation, the reparameterization trick works as follows:

$$w \sim \mathcal{N}(\mu, \sigma)$$

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

In this way the trainable variational parameters μ and σ are separated from the random variable ε as showed in Figure 3.4 and it is possible to optimize the parameters.

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. Figure 3.5 illustrates the training of a Bayesian neural network using different numbers of samples for each update of the parameters. The convergence of the ELBO loss is faster as we use more samples but the value at convergence is approximately the same and the change in convergence speed is very small if compared to the execution time which goes from approximately 22 seconds per epoch when using one sample to more than 2 minutes with 100 samples. For these reasons the we will keep on training the Bayesian model using only one sample per iteration.

A step of the final optimization process can be summarized as follows:

1. Sample $\varepsilon \sim \mathcal{N}(0, 1)$

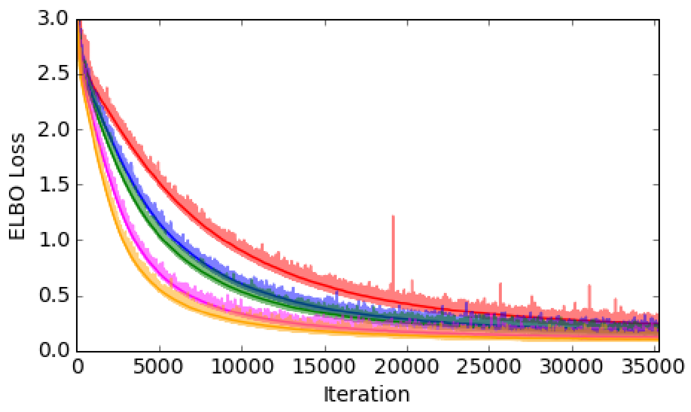


Fig. 3.5 Training of a Bayesian neural network with default prior using respectively 1 (red), 5 (blue), 10 (green), 50 (magenta) and 100 (orange) output samples for each iteration.

2. Let $w = \mu + \sigma\varepsilon$
3. Let $f(w,\theta) = -\log p(\mathscr{D}|w) + \log \frac{q(w|\theta)}{p(w)}$, 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 α:

$$\mu \leftarrow \mu - \alpha\nabla_\mu$$

$$\sigma \leftarrow \sigma - \alpha\nabla_\sigma$$

The update of σ could lead to impossible negative values, therefore it is good practice to reparameterize the standard deviation using a parameter ρ that can assume any value on the real axis:

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

This transformation is called *softmax* and it guarantees that the standard deviation is always going to be positive.

3.2.3 Local reparameterization trick

As already explained, 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 so called local reparameterization trick (Kingma et al., 2015) because it

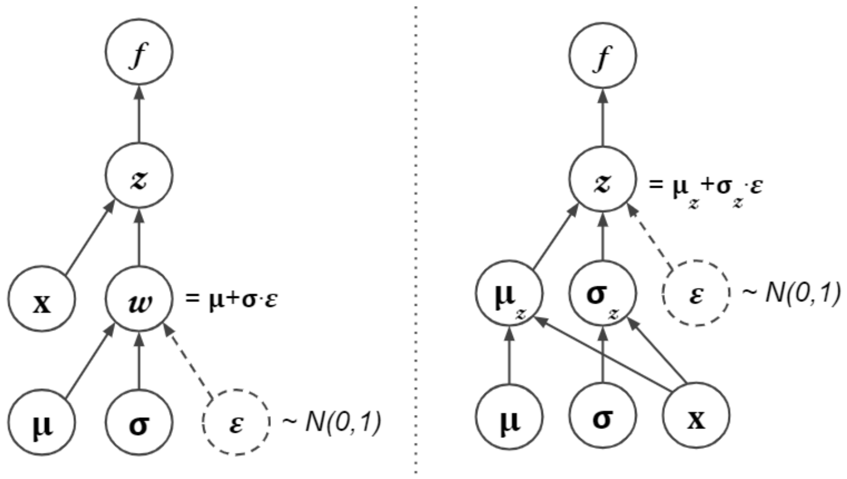


Fig. 3.6 Illustration of the reparameterization trick (left) and the local reparameterization trick (right).

transforms the global noise ε of the model's weights into local noise of the output of each layer ε_z. This process is illustrated in Figure 3.6 using the same scheme as in Figure 3.4.

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:

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

The variational parameters, μ and σ, are used to compute the parameters of the layer activation distribution as follows:

$$\mu_z = \mathbf{X}\mu_w + \mu_b$$

$$\sigma_z = \sqrt{\mathbf{X}^\top \mathbf{X}\sigma_w^2 + \sigma_b^2}$$

This trick has many advantages, in fact, not only it decreases the variance of the gradients, but also it is more computationally efficient because it requires less random samples ε.

Figure 3.7 illustrates the loss of a Bayesian neural network trained using both the standard reparametrization trick and the local reparameterization trick. Since the variance of the gradients is lower, the local reparameterization model decreases faster and converges to a slightly better value.

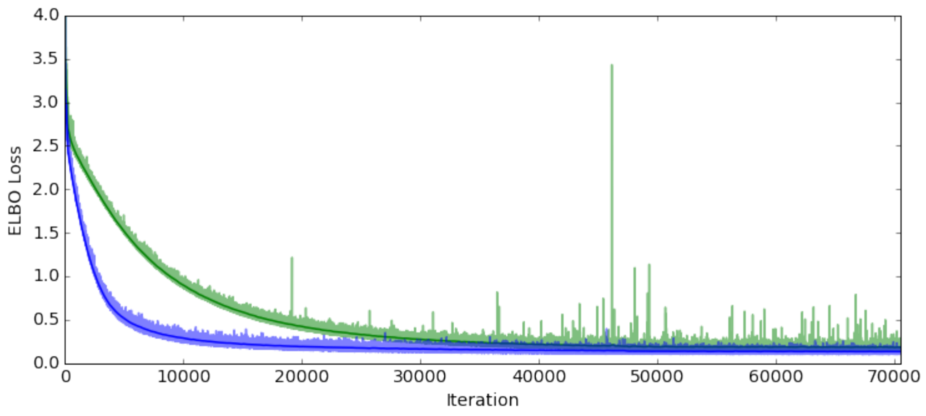


Fig. 3.7 Training loss of a Bayesian neural network using the reparameterization trick (green) and the local reparameterization trick (blue).

3.3 ELBO interpretation

The loss function used to train the Bayesian neural network corresponds to the negative ELBO. A more detailed analysis of the formulation of the variational lower bound leads to a possible interpretation for its two components (Hinton and Van Camp, 1993):

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

The expectation of the log likelihood is a model fit term, given samples from the posterior approximation *q(w)*, it measures how well on average they fit the training data, a higher value means a better fit. The Kullback–Leibler 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.

3.3.1 Regularization using the prior

Moreover, it can be proved that introducing a prior on the model parameters has a regularization effect (Blundell et al., 2015). The Maximum Likelihood Estimates are usually computed as follows:

$$\mathbf{w}^{MLE} = \arg\max_{\mathbf{w}} \log p(\mathscr{D}|\mathbf{w})$$

By adding a prior on the parameters we get a quantity that is proportional to the model posterior, hence the solution becomes Maximum A Posteriori:

$$\mathbf{w}^{MAP} = \arg\max_{\mathbf{w}} \log p(\mathscr{D}|\mathbf{w})p(\mathbf{w}) = \arg\max_{\mathbf{w}} \log p(\mathscr{D}|\mathbf{w}) + \log p(\mathbf{w})$$

Assuming that the prior is a Gaussian $\mathscr{N}(0, \lambda^{-1})$, where $\lambda = \frac{1}{\sigma^2}$ is the precision, the problem can be reformulated in the following way:

$$\mathbf{w}^{MAP} = \arg\max_{\mathbf{w}} \log p(\mathscr{D}|\mathbf{w}) - \lambda \mathbf{w}^\top \mathbf{w} + const.$$

Therefore, ignoring the constant term, the prior term has been transformed into an L2 regularization term. In the same way, it can be proved that a Laplace prior yields L1 regularization.

3.3.2 Prior distributions

The choice for the prior distribution can be trivial for simple models such as linear regression because of the interpretability of model parameters, slope and intercept. 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 choice of a Gaussian prior, together with a fully factorized posterior approximation, has also a computational advantage over more sophisticated priors because the KL divergence can be computed analytically, the closed form solution is the following:

$$KL\{q(w)||p(w)\} = \int q(w) \log \frac{q(w)}{p(w)} dw = \int q(w) \log q(w) dw - \int q(w) \log p(w) dw =$$

$$= -\frac{1}{2}(1 + \log 2\pi\sigma_q^2) + \frac{1}{2}\log 2\pi\sigma_p^2 + \frac{\sigma_q^2 + (\mu_q - \mu_p)^2}{2\sigma_p^2} =$$

$$= \log \frac{\sigma_p^2}{\sigma_q^2} + \frac{\sigma_q^2 + (\mu_q - \mu_p)^2}{2\sigma_p^2} - \frac{1}{2}$$

On the other side, if a closed form solution is not available, it is possible to approximate the expectation using Monte Carlo samples:

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

As already showed, a Gaussian prior implies weight decay with a regularization factor inversely proportional to the variance of the distribution. The variance of the weights distributions 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 parameters get wider distributions. Since the KL divergence can be approximated using samples from q, there is no constraint for the choice of the prior distribution (Ranganath et al., 2014). In particular, a useful prior is the scale mixture distribution with two densities having both zero mean, and variances respectively very small and very big:

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

The parameter π is used to assign different importance to the two components. Figure 3.8 illustrates the Laplace and Gaussian priors used to get the known regularizations, together with the scale mixture distribution and the spike and slab prior (Mitchell and Beauchamp, 1988). As it is possible to see, the scale mixture prior can be used to approximate this latter distribution (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 for greater values because it has heavier tails than a normal distribution. A deeper analysis of the weight distribution will be conducted in Section 4.3.

3.4 Bayesian neural networks training

Stochastic variational inference is well suited for mini-batch optimization which is often used to train neural networks. Mini-batch gradient descent is a tradeoff between standard gradient descent, which requires to compute the loss function over all the observations in the datasets before performing an update of the weights, and stochastic gradient descent, that updates model parameters after looking at each data point (Ruder, 2016). For this reason, depending on the size of the mini-batch, the gradients will have higher or lower variance, influencing the choice of the learning rate as well. The ELBO loss function to train a Bayesian neural network can be defined in the following way:

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

where $\mathcal{D}_i$ represents the i-th mini-batch and β is the weight of the complexity term. The simplest choice for this hyper-parameter would be $\beta = 1/M$, where M is the number of

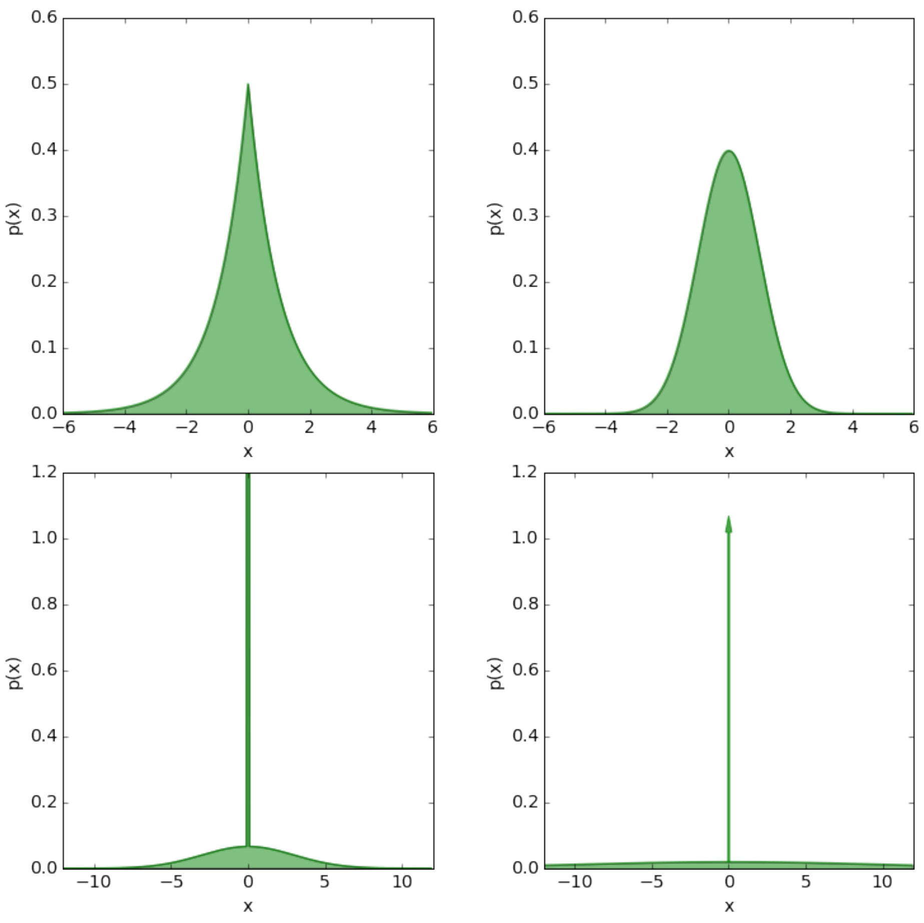


Fig. 3.8 Examples of prior distributions: Laplace distributions (top-left), Gaussian distribution (top-right), scale mixture distribution (bottom-left) and spike and slab (bottom-right).

mini-batches composing the dataset. A more sophisticated solution is to use β so that:

$$\beta_i = \frac{2^{M-i}}{2^M - 1}, \quad i \in \{1, ..., M\}$$

In this way β is not uniform across mini-batches but still $\sum_{i=1}^{M} \beta_i = 1$. It assumes higher values at the beginning of each epoch and it decreases rapidly as illustrated in Figure 3.9.

This approach follows one of the main properties of Bayesian learning, while initially the prior beliefs have high importance, as the model sees more data, the prior becomes less influential.

3.4.1 Scaling the KL divergence

Unfortunately, while in theory the optimization of the defined ELBO loss should lead to the optimal variational parameters, in practice this is not always the case because its formulation suffers of one major limitation. The ELBO loss is composed of two components that depend respectively on the dataset and the model parameters. In particular, the magnitude of the two terms depends on different factors therefore in presence of a very big dataset with very few parameters, the negative log likelihood will start from a very high value while the complexity term will be very small because it is the summation of the KL divergence of each posterior component from the prior. In this case the optimizer will favour optimizing the the likelihood until the it has a similar magnitude to the KL divergence but, depending on the size of the dataset, this could never happen.

Even if this scenario could seem troubled, in theory the presence of a large dataset could overcome the lack of regularization. A more critical and realistic scenario is the opposite one in which the dataset is not very big but the model is very complex, meaning that it has a big amount of parameters. In this case the KL divergence is incredibly higher than the negative log likelihood and, when they reach the same magnitude, the optimizer is not able to start training the parameters but only to keep on reducing the KL divergence.

A good value of β would lead to an initial value of the complexity term that is comparable to the negative log likelihood. Choosing a β that depends on the size of the dataset is only partially correct because the KL divergence need to be scaled with respect to the number of parameters of the model, by doing so, its value would only increase depending on the quality of the variational parameters and not directly with their number. In the same way, the

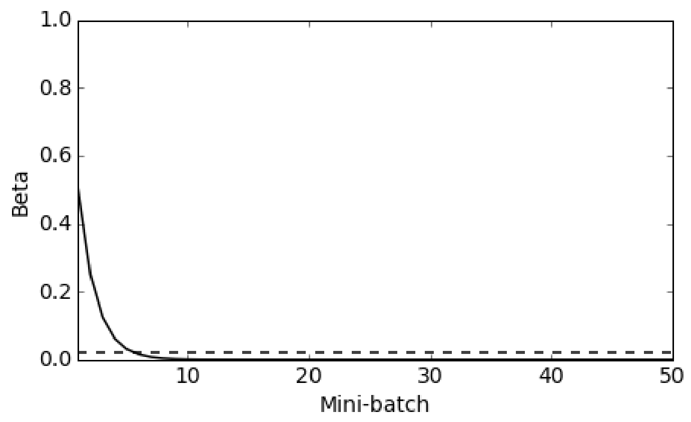


Fig. 3.9 Illustration of the function $\beta = \dfrac{2^{M-i}}{2^M - 1}$ (solid line) and the simpler alternative $\beta = 1/M$ (dashed line) to weight the KL divergence inside the ELBO loss.

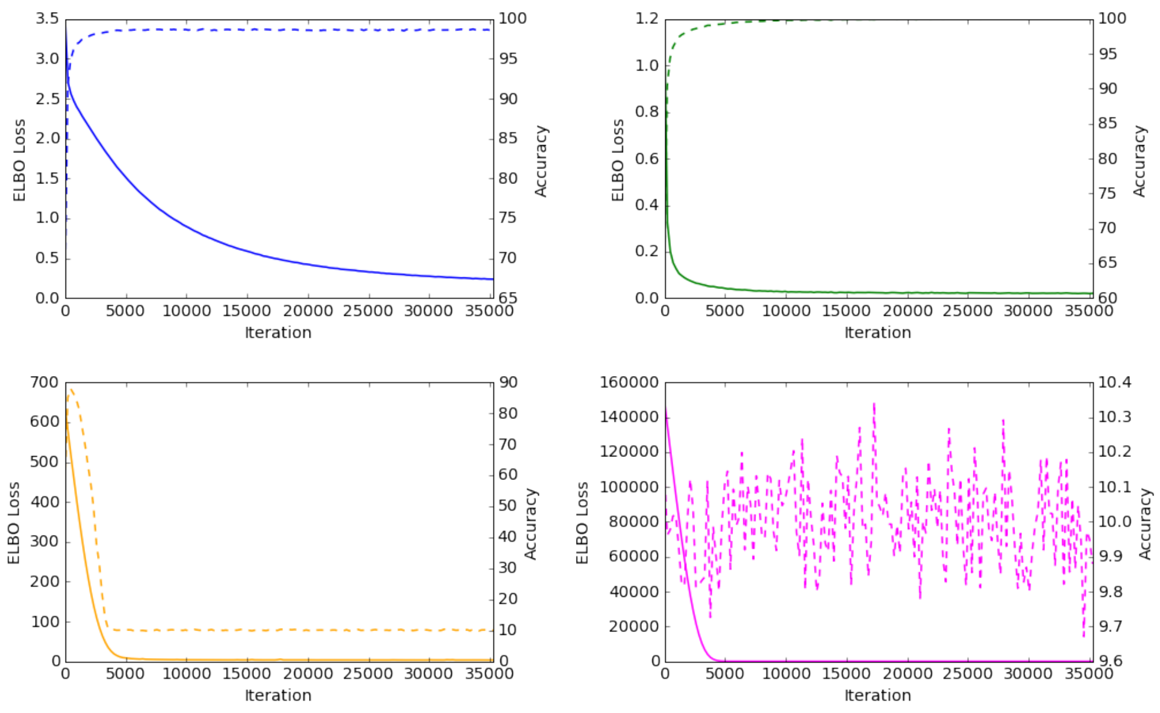


Fig. 3.10 ELBO loss (solid line) and training accuracy (dashed line) for the training of a Bayesian neural network with approximately 60000 weights on a dataset of 60000 images. Four different β are used: (number of weights)$^{-1}$ (top-left), $\frac{2^{M-i}}{2^M-1}$ (top-right), $1/M$ (bottom-left) and 1 (bottom-right).

likelihood needs to be scaled by the number of data points inside the mini-batch so that it varies depending on the quality of the predictions instead of their number.

Figure 3.10 illustrates the ELBO loss and the training accuracy for different values of β. The negative log likelihood is averaged over the data points of each mini-batch. When $\beta = 1/M$ the loss has great values due to KL divergence which is not scaled enough, in fact the accuracy initially increases but then the gradients follow the direction minimizing the KL divergence and the accuracy drops down.

An exasperation of this scenario is the case of $\beta = 1$ where the ELBO loss has a huge value and the parameters are purely optimized to fit the prior distribution leading to the lowest possible accuracy since the beginning of the training.

When $\beta = \frac{2^{M-i}}{2^M-1}$ the results are ambiguous, in fact the loss decreases and the accuracy increases even better than in the case where β is equal to the inverse of the number of weights. This may lead to think that this is the best choice of β but what is actually happening is its value is too small for most of the training as seen in Figure 3.9 therefore the model is optimized without considering the KL divergence, leading to overfitting weights and a bad approximation of the posterior reflecting in the predictive variance more than in the prediction themselves.

It is important to underline that the value of β may only partially improve the quality of the training by scaling the regularization term. In fact, the magnitude of this regularization term mainly depends on the flexibility of the posterior approximation as well as the shape of the prior because these are the factors determining the value of the KL divergence.

Finally, it is worth to mention another possible solution to this problem, called warm-up. It consist in defining a new β^{WU} so that:

$$\beta_i^{WU} = \beta * \gamma_i, \quad \gamma_i = \frac{i}{I-1}, \quad i \in \{0, 1, ..., I-1\}$$

where i is the current epoch and I is the number of initial warm-up epochs. During these epochs the value of $\beta^W U$ increases from 0 to 1 allowing the optimizer to start by optimizing the negative log likelihood without being blinded by huge values of the KL divergence.

3.5 Monte Carlo dropout

At the beginning of this chapter we named MC dropout as another alternative to implement Bayesian neural networks (Gal and Ghahramani, 2016). Dropout itself has been initially introduced has a regularization technique for standard neural networks (Hinton et al., 2012). It works by randomly turning off the activation of each layer as illustrated in Figure 3.11. In this way, the weights are trained to be more robust because less dependents on each other.

In order to make predictions all the weights are scaled by the dropout probability, that is the probability that they are turned off during the training. Monte Carlo dropout instead,

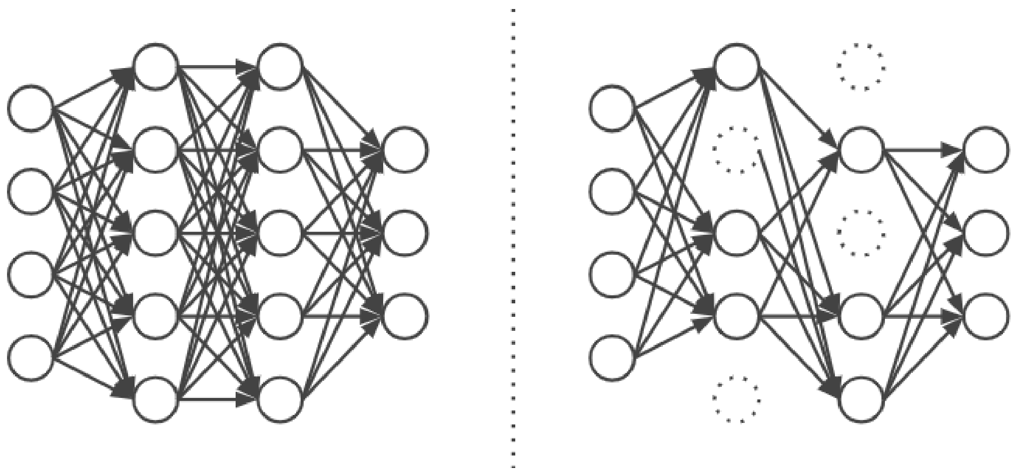


Fig. 3.11 Illustration of the dropout technique.

keeps the dropout rate also at prediction time, and then averages the results over a number of stochastic forward passes in the same way as it has been showed in Section 2.4.2.

For these reason, MC dropout, is able to build a predictive distribution and therefore provide a predictive uncertainty, in fact it is equivalent to minimizing the KL divergence between an approximating distribution and the posterior of a deep Gaussian Process (Gal, 2016), that is a stack of Gaussian Processes connected in a neural network fashion. In particular, the approximating distribution works under the previously mentioned spike and slab prior, in fact for each sample, that is for each iteration, the weights values are either zero or the regularized value.

Chapter 4

Uncertainty analysis

We will conduct the first uncertainty analysis using a Bayesian neural network with a known architecture on a simple dataset. The MNIST dataset (LeCun and Cortes, 1998) is made of images of single digit numbers and the corresponding label as illustrated in Figure 4.2.

We decided to work with this data because it is easy to interpret, in fact a human can immediately tell if a given image corresponds to the associated label and it is also possible to distinguish between well written digits, bad written ones and ambiguous ones. Moreover, it is also possible to use other interesting datasets like EMNIST (Cohen et al., 2017) or notMNIST (Bulatov, 2011) to benchmark the capability of the model to find out-of-distribution samples.

The neural network architecture is the same as LeNet5 (LeCun et al., 1998) because of its simplicity and its very good performance, even without any kind of tuning (LeCun et al., 1995). It has been used in its most naive version, illustrated in Figure 4.1

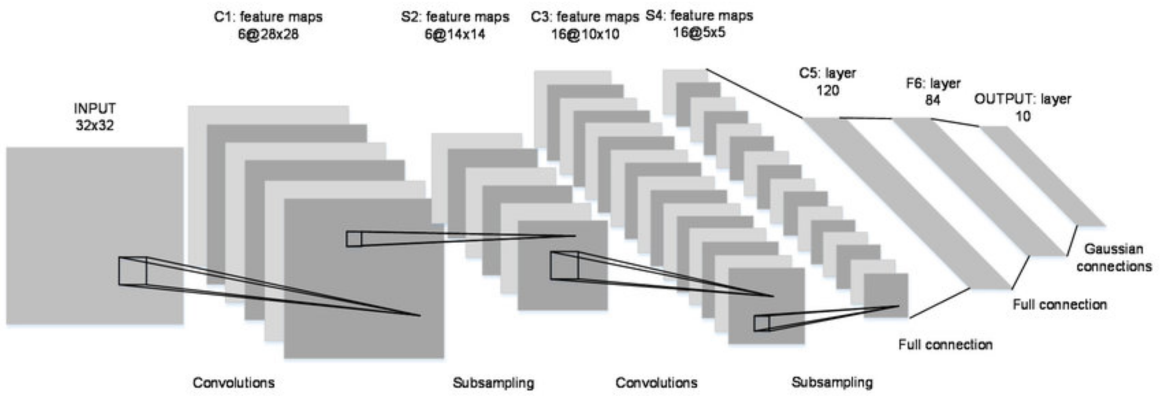


Fig. 4.1 Architecture of the convolutional neural network LeNet5 from (LeCun et al., 1998)

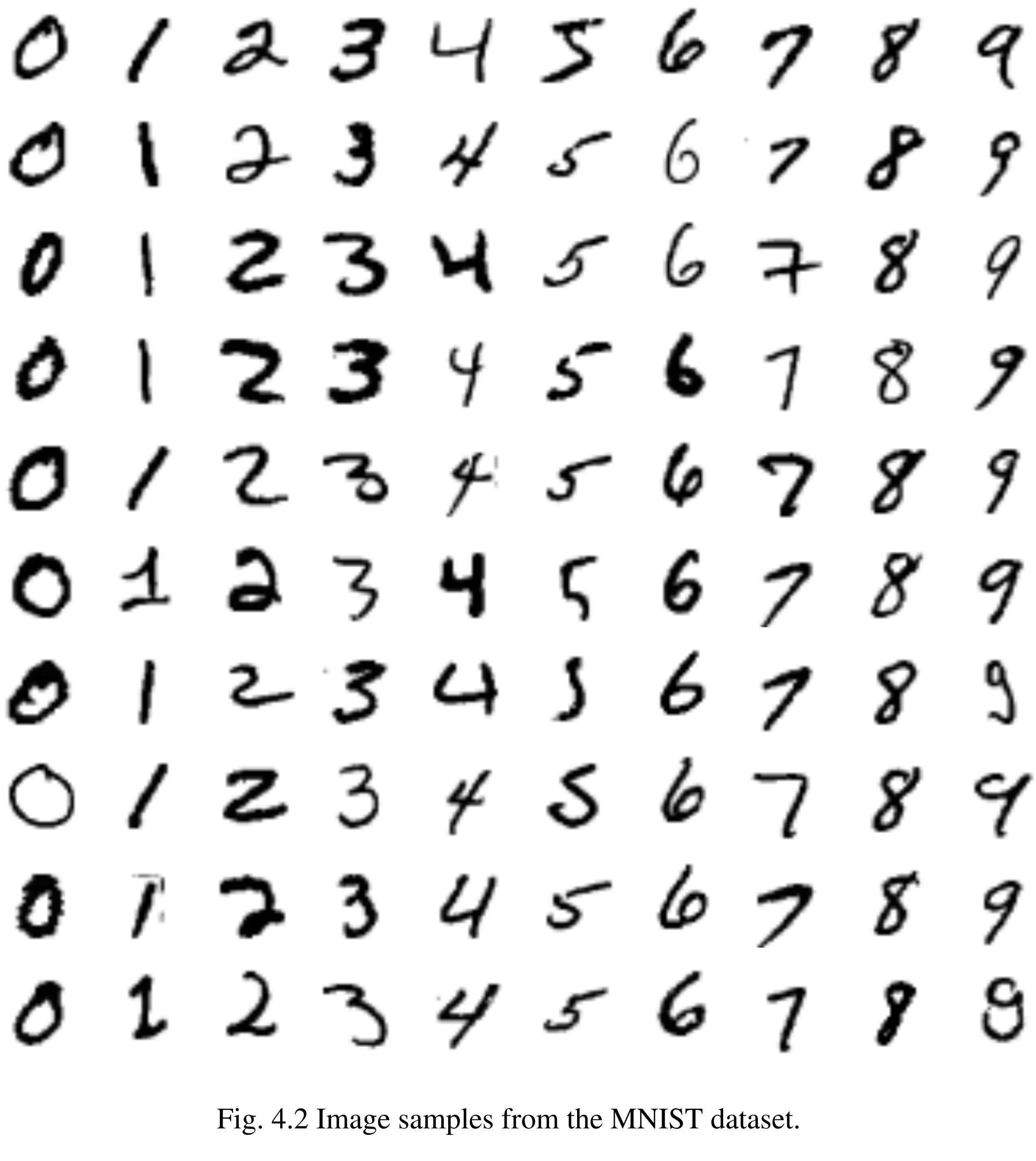


Fig. 4.2 Image samples from the MNIST dataset.

4.1 Training analysis

Over the previous chapters we introduced the differences between the standard neural networks using point estimates of the weights and the Bayesian ones. The main differences between these two approaches start appearing during the training.

As already stated in Chapter 3, training a Bayesian model (using a normal mean field approximation) with the same architecture as a standard one, means to train a model with twice the number of parameters because instead of optimizing a point estimate for each weight of the neural network, the mean and standard deviation of the corresponding Gaussian distribution are considered.

We expect the Bayesian model training to be more time consuming because of the higher number of parameters and also because the optimizer needs more time to find a local minimum in a space with higher dimensionality.

4.1.1 Standard neural network training

Figure 4.3 illustrates the training of a standard implementation of LeNet5 on the MNIST dataset. This network has been trained using the Adam optimizer (Kingma and Ba, 2014) and a simple learning rate schedule consisting of a single step decreasing the learning rate by a factor of 10, starting from 10^{-3}. Moreover we applied the widely used cross entropy loss function which integrates perfectly with the softmax output of a neural network learning a classification task. Finally, we also 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 this dataset and a wide number of further improvements could have been used (e.g. Dropout (Hinton et al., 2012),

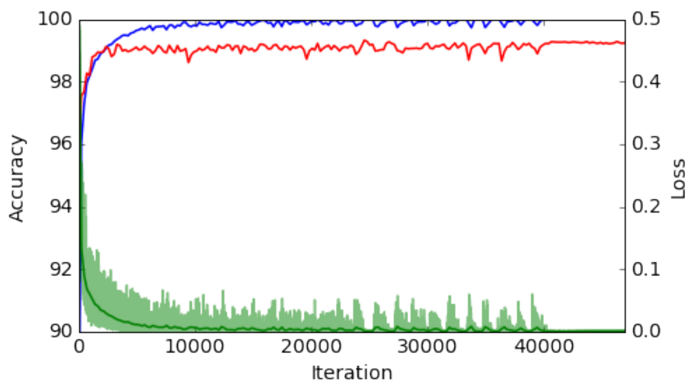


Fig. 4.3 Loss (green), training accuracy (blue) and test accuracy (red) during the training of LeNet5 on the MNIST dataset.

Batch Normalization (Ioffe and Szegedy, 2015), Data augmentation (Simard et al., 2003)). This implementation choice has been taken because the model is still able to perform very well as illustrated by the final accuracies (100% on training data and 99.32% on validation data) and, since the dataset is contextually simple, further improvements could lead to higher accuracies but complicate the comparison with the Bayesian counterpart.

The network has been trained for a total of 200 epochs, the accuracy and the likelihood converge in a few epochs and they finally stabilize when the learning rate decreases to 10^{-4} at epoch 170.

4.1.2 Bayesian neural network training with default prior

Figure 4.4 shows the training on the same dataset of a Bayesian neural network using the default prior distribution: a Gaussian with zero mean and unit variance.

Given the different nature of this model with respect to the previous one, the training curves look different too. This model has been trained for 300 epochs with a single learning rate step at 200 epochs, starting from 10^{-3}. The loss function is the weighted sum of the previously used cross entropy loss and the KL divergence, namely the ELBO loss. No weight decay has been applied because the regularization is already imposed by the shape and parameters of the prior.

The accuracy of this model does not converge to the same value as the previous one. This is due to the fact that the performances illustrated in this plot are computed during the training using a single sample from the weight distribution, however, the final accuracy will also be lower than the standard model one.

Another symptom of the sample-based computation of these metrics is the variance of the ELBO loss which does not decrease during the training.

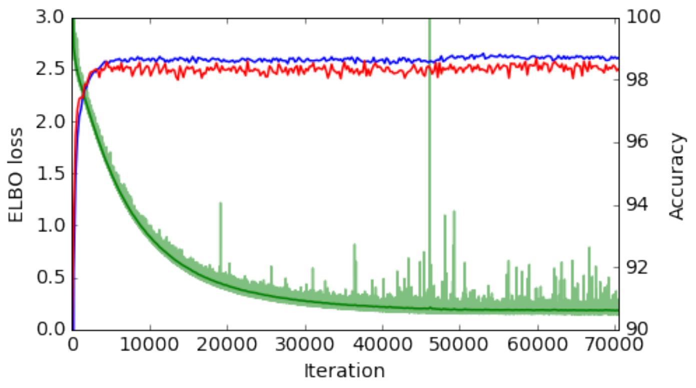


Fig. 4.4 Loss (green), training accuracy (blue) and test accuracy (red) during the training of Bayesian LeNet5 on the MNIST dataset using the default prior distribution.

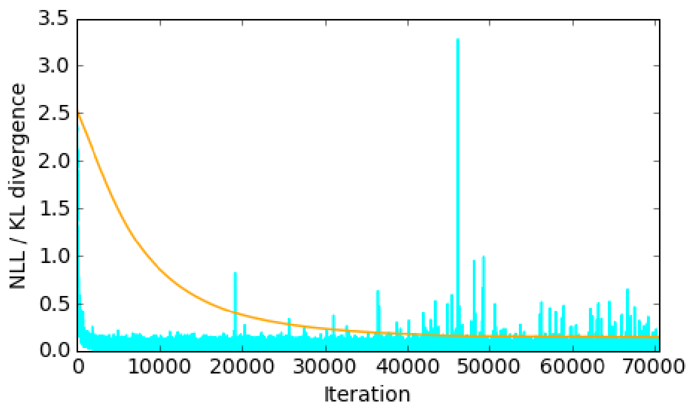


Fig. 4.5 Negative log likelihood (green) and scaled KL divergence (orange) of a Bayesian neural network during the training.

Moreover, the ELBO loss decreases more slowly compared to the previous model and has some big spikes starting from the middle of the training.

In order to understand this behaviour we analyzed the single components of the ELBO loss as illustrated in Figure 4.5. This plot shows that, while the negative log likelihood decreases immediately and reaches a value comparable to the one of the standard model, the KL divergence is the component making the ELBO loss decrease more slowly. Furthermore, the spikes in the loss are generated by the negative log likelihood and they start to appear when the KL divergence reaches smaller values. This two phenomena are strictly linked because a lower KL divergence implies that the distribution of more parameters in the model is similar to the prior, which has a big variance. Therefore, sample from these distributions will be more heterogeneous and lead to different predictions. The final weight distributions will be analyzed in more detail in Section 4.3.

4.1.3 Bayesian neural network training with scale mixture prior

Figure 4.6 illustrates the same training as previously done but with a scale mixture prior. As discussed in Section 3.3, the choice of the prior distribution can have a big impact on the final results and also on the training.

The main difference lays in the final value of the ELBO loss which, again, is mainly due to the KL divergence because the negative log likelihood always converges to approximately zero. The reason for the KL divergence to be bigger than before is simply due to the shape of the posterior compared to the prior, while the Gaussian posterior can easily fit the default prior because they are both normal distributions, the scale mixture prior always leads to a higher KL divergence because it cannot be completely fit by a single scale normal distribution.

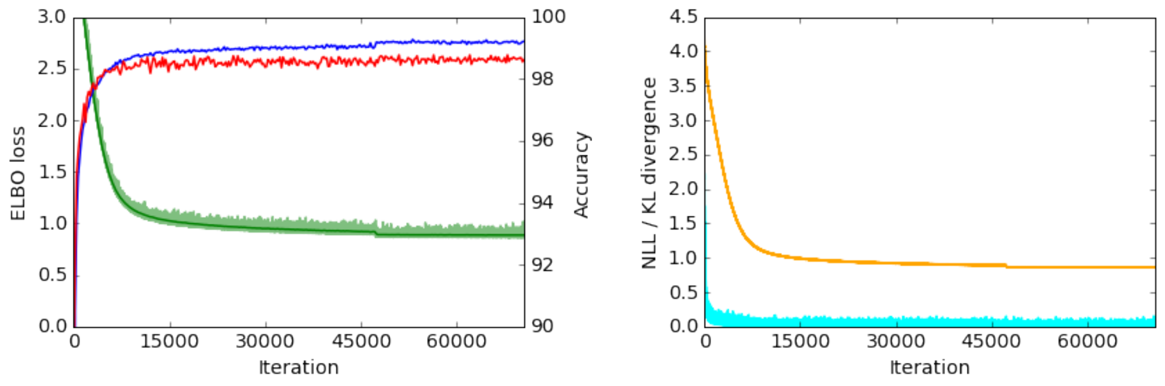


Fig. 4.6 Training of a Bayesian neural network with scale mixture prior. ELBO loss (green), training accuracy (blue) and test accuracy (red) on the top, negative log likelihood (green) and scaled KL divergence (orange) on the bottom.

It is also possible to appreciate less oscillations in the negative log likelihood after the KL divergence decreases, this is due to the better discrimination of the scale mixture prior between useful weights that become more spiky and useless ones, that become flatter.

4.1.4 Bayesian neural network training with the local reparameterization trick

Finally, Figure 4.7 shows the training of a Bayesian neural network using the reparameterization trick and both the default prior and the scale mixture prior.

In Section 3.2.3 we mentioned that the local reparameterization trick decreases the variance of the gradients, and allows a faster convergence of the parameter values.

Independently from the prior used, the ELBO loss converges faster than before. Furthermore, the local reparameterization trick impacts also the variance of the activations of each layer, allowing the ELBO loss to be more stable, in fact the neural network with default prior does not show spikes anymore.

The comparison of the variance of the gradient norm during the training of two Bayesian neural networks with default prior and scale mixture prioris illustrated in Figure 4.8.

4.2 Accuracy analysis

As already outlined in Section 4.1.2, the true accuracy of the model is not the one computed during the training because the predictions are calculated using a single sample from the posterior distribution. In Section 2.3.3 we showed the integral to compute the predictive

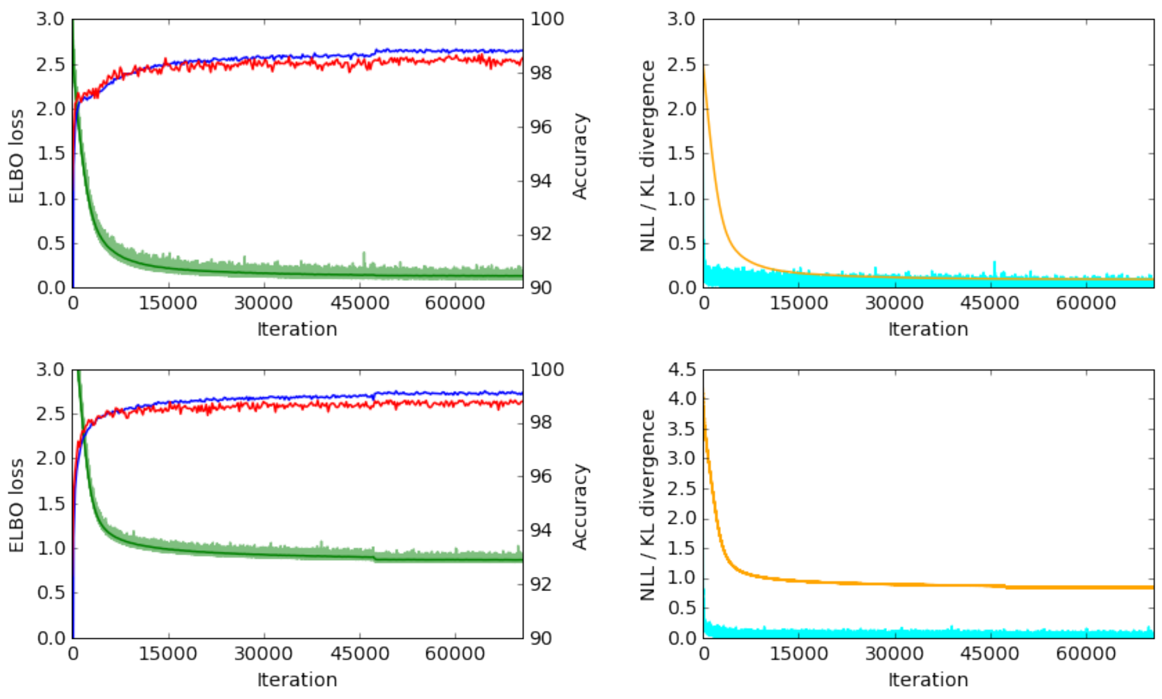


Fig. 4.7 Training of a Bayesian neural network with default prior (left) and scale mixture prior (right) using the local reparameterization trick. ELBO loss (green), training accuracy (blue) and test accuracy (red) on the top, negative log likelihood (cyan) and scaled KL divergence (orange) on the bottom.

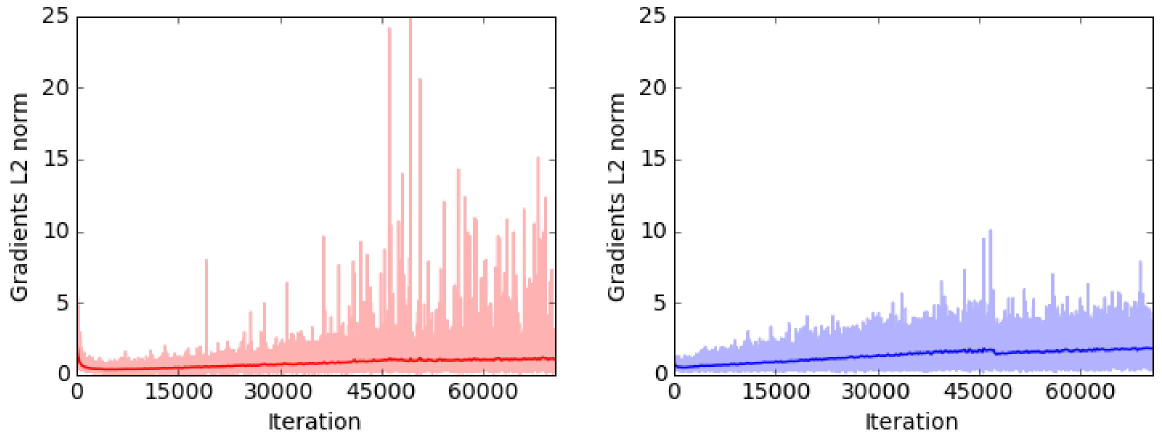


Fig. 4.8 L2 norm of the gradients of the weights of a Bayesian neural network with default prior trained using only the reparameterization trick (left) and with the local reparameterization trick (right).

distribution and in Section 2.4.2 we approximated the same integral via Monte Carlo sampling. In order to do so it is needed to sample multiple times from the posterior distribution, compute the output for each sampled set of weights and average over all the outputs.

Figure 4.9 shows the output counts for an image of a four, each sampled output is used to build the histograms over the different classes. Since this image is ambiguous, in fact it looks like a nine, it is possible to see that some outputs predicted a nine instead of a four. If the same experiment would have been done with a standard neural network, the corresponding histograms would have showed no uncertainty because the output of the model, given a certain input, is always the same.

Another visualization of model uncertainty is proposed in Figure 4.10. On the left, 100 predictions obtained by different samples from the weight distribution are displayed in different colors. Some of them are always predicted with the same label, independently from the particular weights sample, in fact the corresponding column in the figure is all in the same color. Other images instead, are assigned to different classes depending on the weight sample, therefore the corresponding column contains multiple colors. These are the images we expect to be more noisy or ambiguous (as the one previously displayed), hence they will have higher predictive uncertainty.

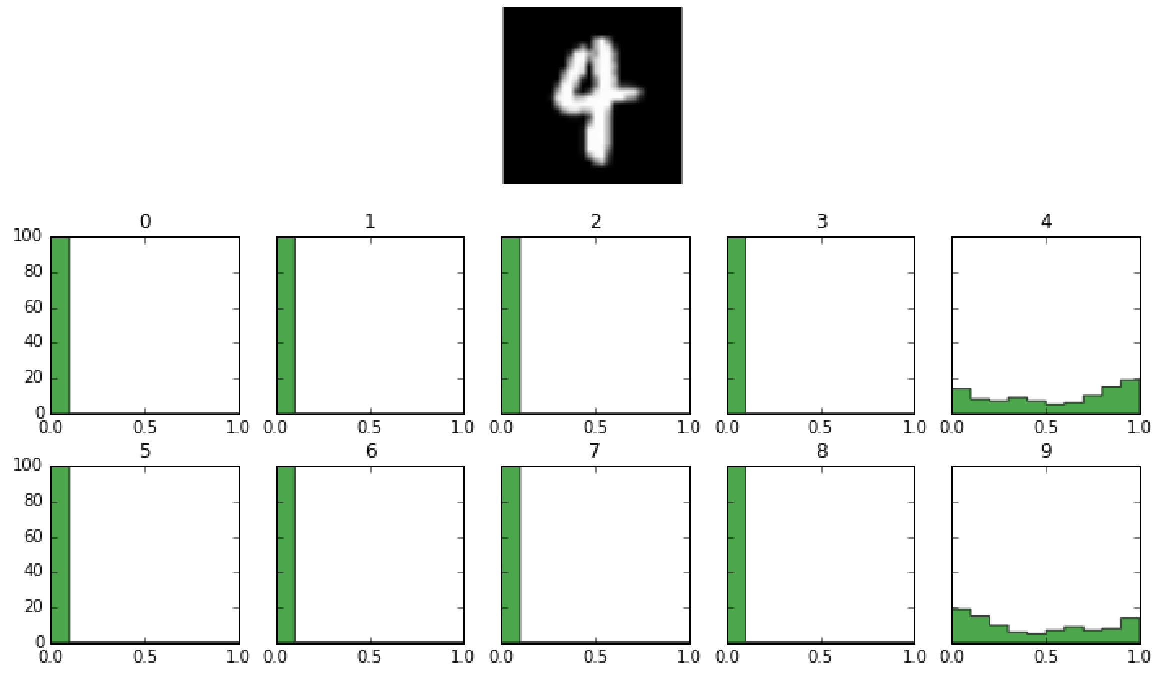


Fig. 4.9 Image of a 4 and histograms for each class corresponding to 100 outputs of a Bayesian neural network when given this image as input.

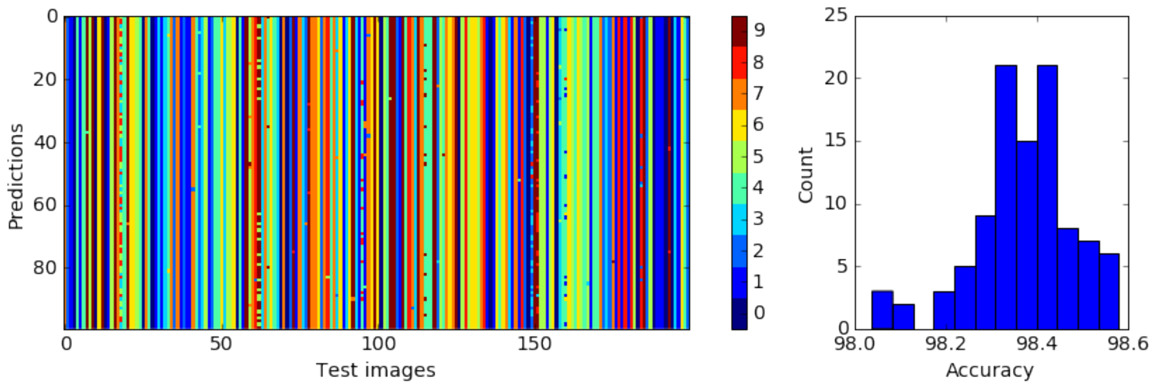


Fig. 4.10 100 predictions of 200 test images from the MNIST dataset using a Bayesian neural network (left) and histogram of the corresponding accuracies using all the 10000 test images.

The right side of Figure 4.10 shows the histogram of 100 accuracies computed on one output of the Bayesian neural network, that is by using only one sample from the posterior distribution. The variance of the histogram of the accuracies also suggest there is general uncertainty in the model.

If these figures would have been created using predictions from a standard neural network, all the columns on the left side would have had the same color and the histogram on the right would have had only one value with count equal to 100.

Depending on the number of outputs averaged to compute the final prediction, the corresponding accuracy can be more or less accurate. Figure 4.11 shows 100 accuracies obtained by using different numbers of samples 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 to the real value and the variance of the final accuracy decreases.

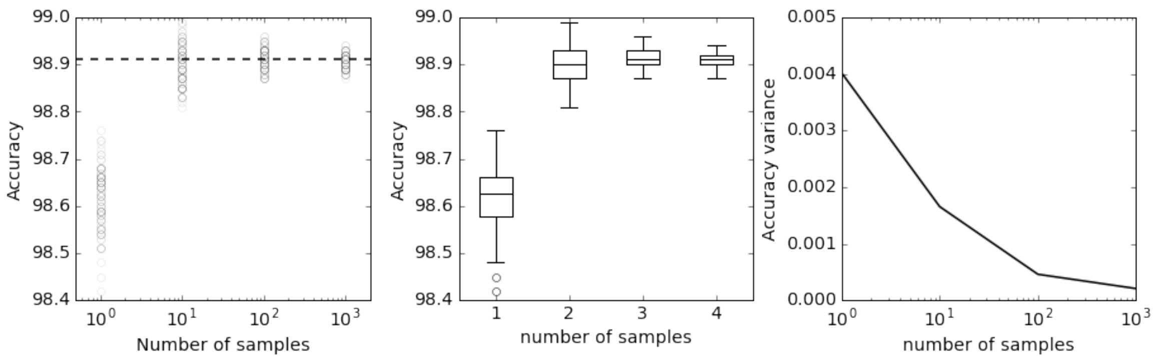


Fig. 4.11 Scatter plot of 100 values of the accuracy of a Bayesian neural network per different number of samples (left), the corresponding boxplot (center) and the variance per number of samples (right).

In the following part of the thesis we will compute predictions and accuracy using 100 samples because the corresponding variance of the accuracy 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, while computing 1000 outputs for a single image takes approximately 1.2 seconds, it only takes approximately 0.1 seconds to get 100.

Finally, Table 4.1 illustrates the accuracy for all the models considered so far. We may decide to select the model with the higher accuracy in order to proceed with the dataset analysis but the focus of this work is not on the accuracy (otherwise we would use a standard neural network) but rather on the uses of the predictive uncertainty, hence we will focus more on the quality of the latter.

	Standard	**Default prior**	**Scale mixture prior**	**Default prior and local rep. trick**	**Scale mixture prior and local rep. trick**
Training	100%	99.62%	99.59%	99.55%	99.11%
Test	99.32%	99.23%	99.13%	99.03%	99.07%

Table 4.1 Training and test accuracies of different neural networks on the MNIST dataset.

4.3 Parameters analysis

The predictive uncertainty comes also from the model parameters, which have been modeled as normal distributions. Therefore the analysis of these distribution can give useful insights before looking at the uncertainty in the predictions

A simple way to analyze the weights of a standard neural network is by looking at the distribution of the values because the lower they are, the less useful, in fact a weight with value zero is equivalent to a pruned weight /todocite.

4.3.1 Standard neural network weight distribution

Figure 4.12 illustrates the histogram of the weights of the standard neural network previously used, most of the weights have a value very close to zero, therefore their activations are probably going to be close to zero as well.

In order to understand the importance of the weights based on their magnitude we computed the accuracy of the neural network after setting to zero all the weights which logarithm of the absolute value is below a certain threshold.

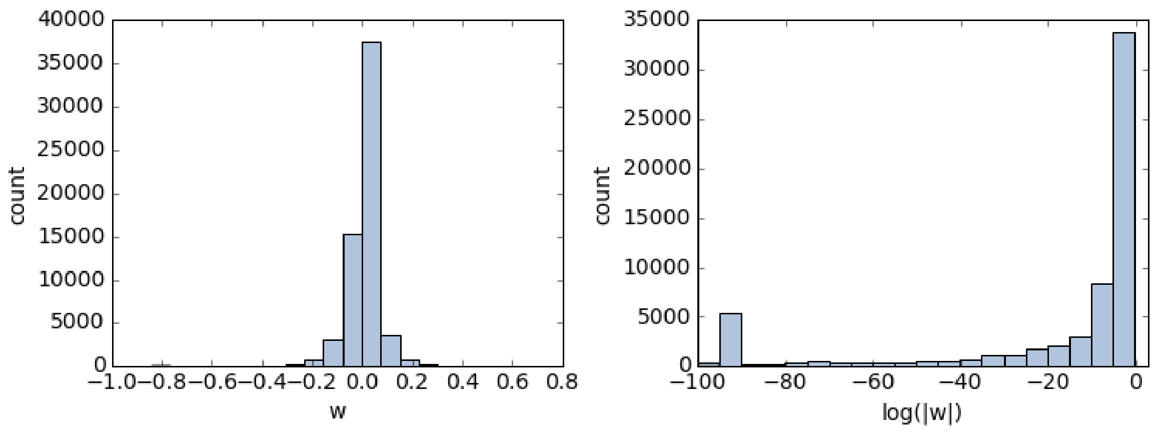


Fig. 4.12 Histogram of the weights of a standard neural network (left) and histogram of the corresponding logarithm of the absolute value (right).

We present in Figure 4.13 the results obtained with different increasing thresholds. In particular, the left plot shows how the accuracy of the network changes with the percentage of parameters pruned. The performances of the network are constant if we remove almost 60% of the weights with low magnitude, then they decrease slightly until approximately 75% of weights are removed and they finally decrease towards lower accuracies.

The right plot of Figure 4.13 shows the accuracies corresponding to the different thresholds used to truncate the weights. The value corresponding to the initial decrease in performance is approximately -4. This value does not give any insight because when placed in the weight distribution (Figure 4.12 (right)) it does not split weight into any relevant subgroup but it falls in the middle of the main group of weights.

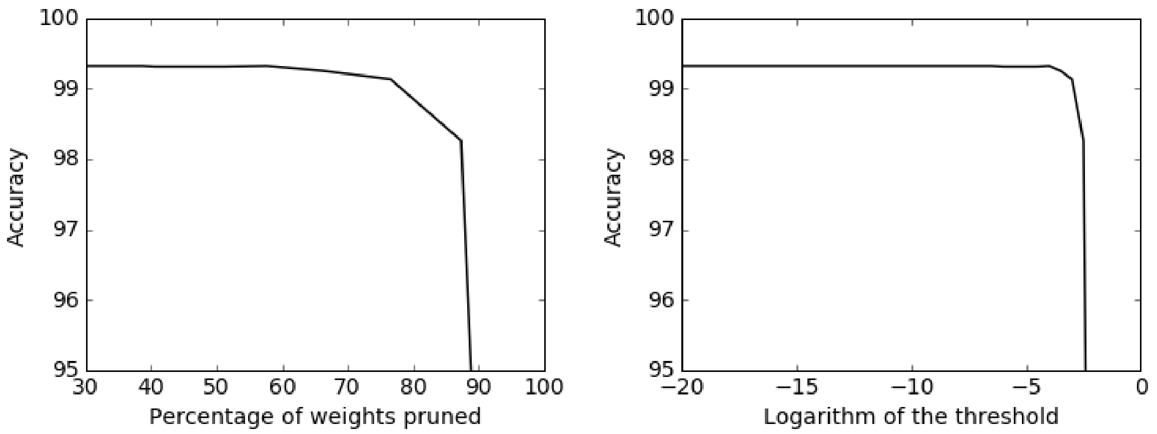


Fig. 4.13 Accuracy versus percentage of parameters pruned (left) for a standard neural network and accuracy versus the corresponding threshold used to prune the weights (right).

4.3.2 Bayesian neural network with default prior parameter distribution

A similar analysis can be applied to the Bayesian neural networks to have a measure of the quality of the weights and the training process.

Since the weights of a Bayesian neural network are not point estimates but distributions, instead of representing them using only their means it is more informative to use the signal to noise ratio which uses both the mean and variance of each distribution (Blundell et al., 2015). It is defined as follows:

$$\mathrm{SNR}(w_i) = \frac{|\mu_i|}{\sigma_i}$$

The SNR distribution and its logarithm are presented in Figure 4.14. It is possible to appreciate a strong similarity between the logarithm of the signal to noise ratio and the previous distribution, due to the fact that, as already discussed in Section 3.3, optimizing the ELBO using a normal prior is equivalent to optimize the likelihood function with L2 regularization. The SNR is not only low when the mean of the distribution is close to zero, but also when it has a large variance. If a weight value changes too much between different samples but the model has still good performance it means that its value is not too relevant to the final output, hence it can be discarded. For this reason it is good to truncate weights depending on their SNR instead of just by using their means.

Even if the distribution of the model's parameters may look similar to the previous one, there are some key differences that can be observed in Figure 4.15. In this case, in order to truncate a weight, both the mean and the standard deviation of the weight distribution

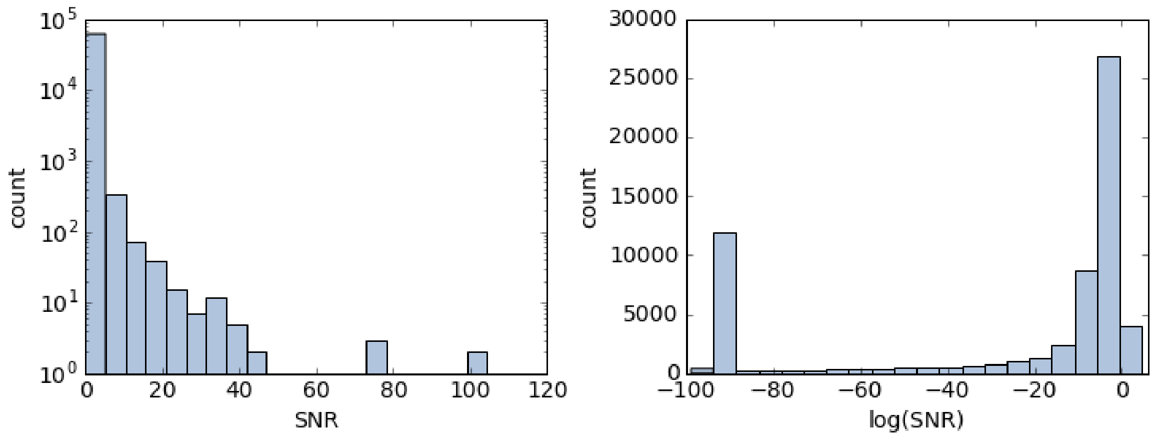


Fig. 4.14 Histogram of the Signal-to-Noise ratio of the parameters of a Bayesian neural network (left) and histogram of the corresponding logarithm (right).

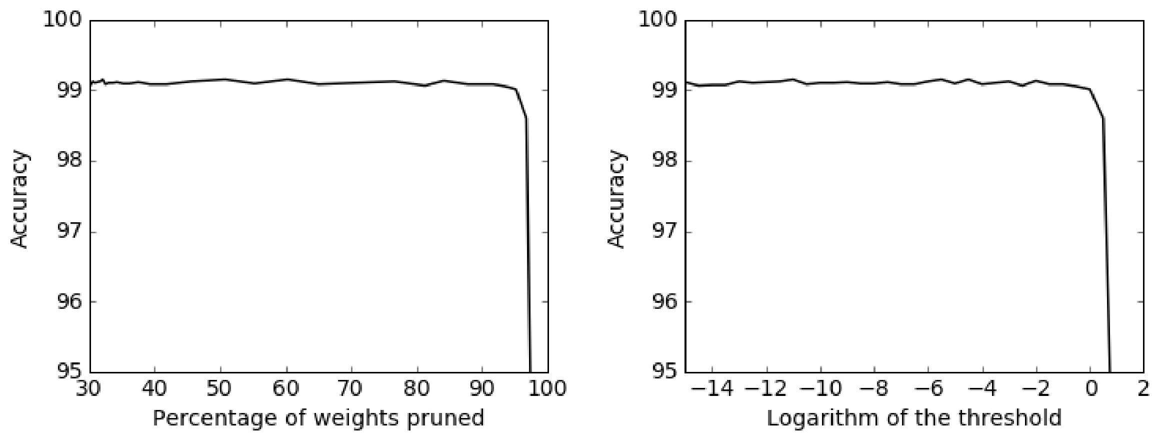


Fig. 4.15 Accuracy versus percentage of parameters pruned (left) for a Bayesian neural network with default prior, and accuracy versus the corresponding threshold used to prune the weights (right).

were set to zero. By comparing this plot to the one in Figure 4.13 it can be noticed that, in the Bayesian model, the accuracy fluctuates. This is not due to the truncation of model parameters but to the stochasticity of the predictions used to compute the final accuracy.

Moreover, the model keeps its original accuracy even after almost 95% of the parameters have been set to zero which is way higher if compared to the original MLE model which performances degrade after truncating 60-80% of the parameters. This result shows how Bayesian models are better regularized and the sparsity of the parameters leads to sparsity in activations and intermediate features which can be better exploited for interpretation of the model and its performance.

4.3.3 Bayesian neural network with scale mixture prior parameter distribution

We already underlined the importance of the scale mixture prior compared to the default one because of the possibility to better regularize and therefore discriminate between parameters with different importance, as can be seen by looking at Figure 4.16, where the number of parameters with mean close to zero is higher than in the previous case, suggesting that more weights can be truncated.

Moreover, the histogram of the logarithm of the SNR shows that the distribution has two main modes. While Figure 4.14 shows a smaller mode which has no particular relevance, by looking at the thresholds position in the histogram of Figure 4.16 we can see how the value corresponding to the change in accuracy (~ 0.5) divides the parameters distributions in two

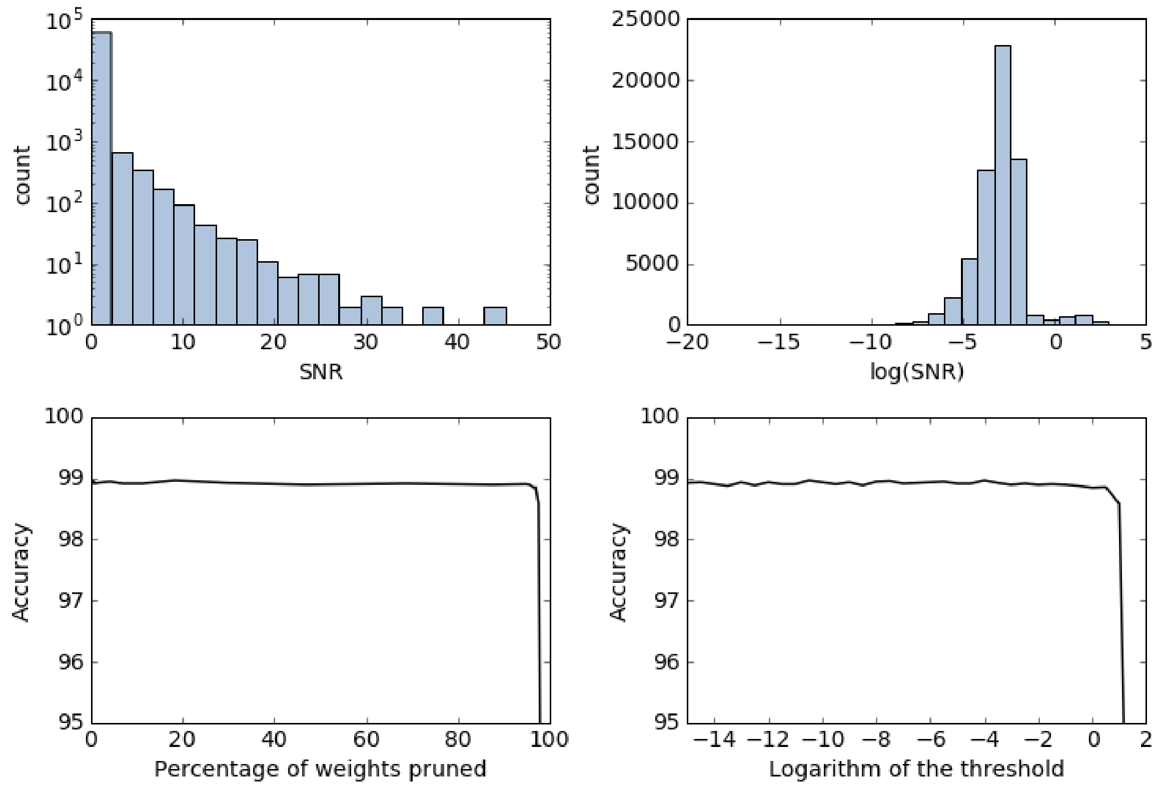


Fig. 4.16 Histograms of the Signal-to-Noise ratio (top) and accuracy change by pruning model parameters (bottom) for a Bayesian neural network with scale mixture prior.

sets, therefore the smaller mode contains the most relevant parameters, confirming that this particular prior better discriminates between useful and useless parameters.

4.4 Uncertainty definition and evaluation

The most important feature of Bayesian models is the possibility to measure uncertainty over the parameters and the predictions. The latter can be defined as the variance of the predictive distribution which has been approximated using Monte Carlo sampling. Since, the output of a classifier has N components, one for each class, uncertainty is defined by the covariance matrix. This measure only considers one of the two types of uncertainty that have been discussed in the introduction to this thesis. In order to get a measure of both aleatoric and epistemic uncertainties we can modify the previous definition in the following way (Kendall and Gal, 2017; Kwon et al., 2018):

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

where $\bar{p}=\frac{1}{T}\sum_{t=1}^{T}\hat{p}_t$, $\hat{p}_t$ is the output of the neural network after the softmax function has been applied and T is the number of output samples taken.

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 may be interesting to look at because they may inform about relations between classes, hence they can be helpful to find ambiguous images or labels.

In practice these elements are most of the time zeros or very low and it is hard to extract any kind of useful information from them, therefore only the components on the diagonal are considered as uncertainty measures.

Furthermore, two approaches may be adopted to extract a confidence measure from this variances vector. The predictive uncertainty can be computed either as the sum of the predictive variances of each class, or just considering the variance of the predicted class, which is the one with the highest mean across multiple samples.

The latter method has been adopted for the following analysis because of its ease of interpretation, a high predictive variance corresponds directly to high uncertainty for the prediction of that specific label. By using the sum of the variances, a high predictive variance may be due to multiple classes as well as a single one which may also happen not to be predicted one. Therefore the predictive uncertainty can be computed in a simplified way, considering only the predicted class variance:

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

The quality of the predictive uncertainty can be determined by analyzing its capacity to discriminate between good and bad predictions. Figure 4.17 shows the predictive variance obtained using the default prior over the MNIST test set of 10000 images. Most of the images have been predicted correctly and with low variance while the wrong predictions are mainly made with high uncertainty. This means that the model is generally certain about its good predictions and it knows when it could be making a wrong one.

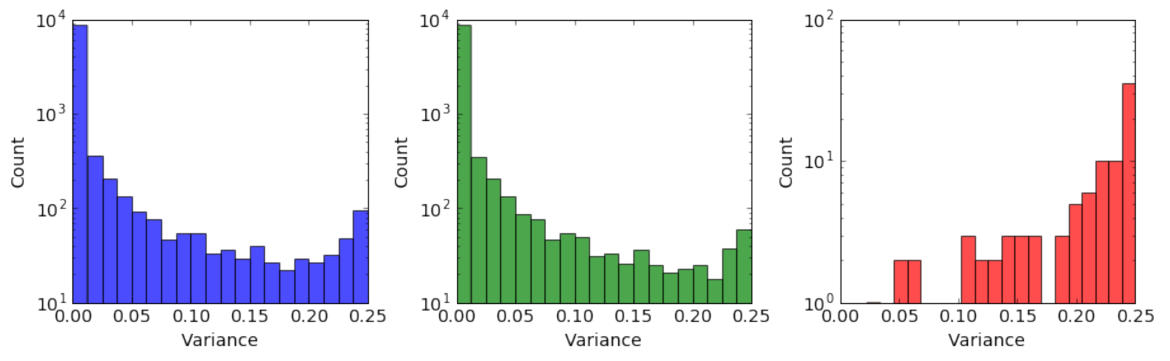


Fig. 4.17 Histograms of the predictive variance of a Bayesian neural network with default prior. The histograms are build using all the test images (left), the correctly predicted images (center) and the wrongly predicted ones (right).

Furthermore, it is worth to underline that some of the correctly predicted images have high predictive variance while very few wrongly labeled images have low variance too. This phenomenon may not be casual and looking at the corresponding images can offer good insights about the dataset and model uncertainty. This analysis will be performed in Section 4.5.

Figure 4.18 illustrates the predictive variance of the model trained using a scale mixture prior. These histograms look slightly better than the previous ones but they are overall almost identical. This suggest that both the models made the same mistakes hence they are uncertain about the same images.

In order to better understand the advantages of Bayesian neural networks over standard ones, we compared the value of the softmax output of the two approaches. As previously stated, the softmax output is only modeled to look like a distribution and does not represent a

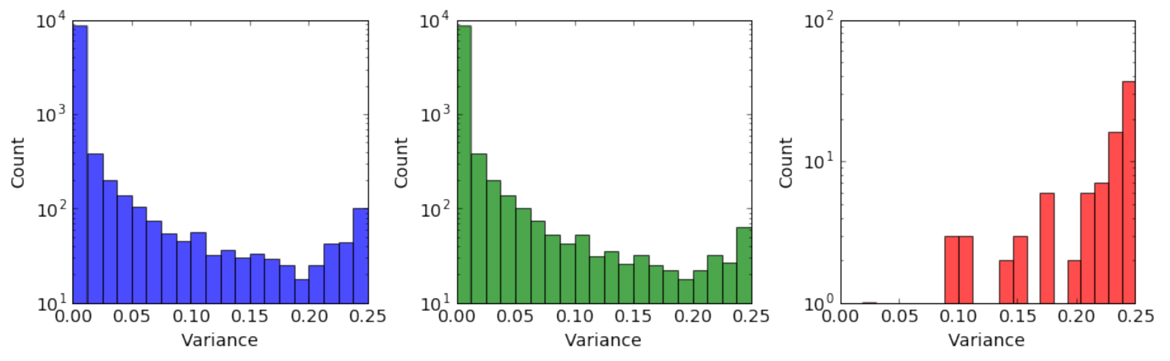


Fig. 4.18 Histograms of the predictive variance of a Bayesian neural network with scale mixture prior. The histograms are build using all the test images (left), the correctly predicted images (center) and the wrongly predicted ones (right).

good uncertainty measure but can still be used to try to discriminate good predictions from bad ones.

Figure 4.19 illustrates the distributions of the softmax output maximum values for the test images of the dataset. Most of the predictions have a low variance, hence they also have a very high corresponding output, both for the standard neural network and the Bayesian one. The main difference between the two models is the range of values covered by the output maximum value: the standard neural networks tend to have outputs with grater values than the Bayesian ones, that are more distributed. For this reason most of the bad predictions of the former model have a high output while the Bayesian ones have lower values. This proves that the softmax output of the standard neural network is not able to provide any uncertainty regarding the predictions.

The final step to measure the benefits of the Bayesian framework is to use the predictive uncertainty to improve the accuracy of the model. For example, if the classifier needs to diagnose cancer by looking at MRI images, this could offer the possibility to pass all the uncertain predictions to an expert and preserve all the correct and certain ones. In order to do so it is needed to discard all the predictions that come with uncertainty higher than a defined threshold and measure the accuracy while changing the threshold value. The same approach

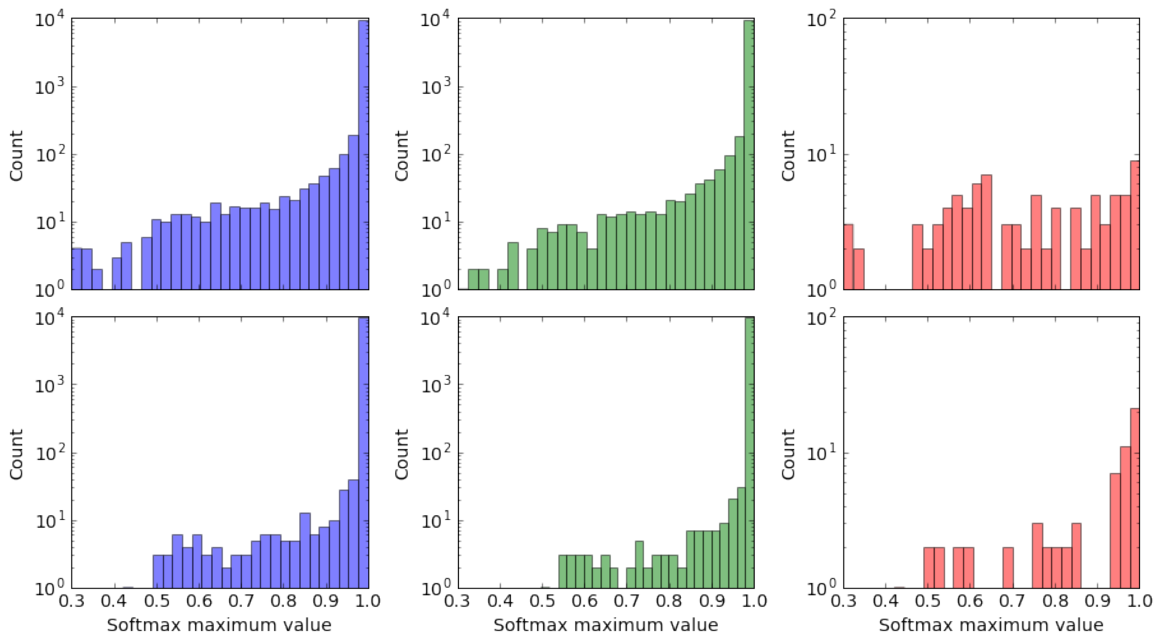


Fig. 4.19 Histograms of the maximum values of the outputs of a Bayesian neural network (top) and a standard one (bottom) for the MNIST test images. The histograms are build using all the test images (left), the correctly predicted images (center) and the wrongly predicted ones (right).

can be applied to the output of the neural network by discarding all the images whose output maximum value is smaller than the threshold. In this way it is possible to further compare the two approaches used so far.

Figure 4.20 shows the accuracy of the model and the corresponding percentage of discarded predictions for different thresholds. The blue and the yellow lines are almost overlapped, hence suggesting a very high correlation between the value of the output of the Bayesian model and the corresponding uncertainty.

The accuracy corresponding to no predictions discarded is the initial accuracy of the models, respectively 99.32% and 99.07% for the standard and Bayesian neural networks. After discarding the initial predictions both the models improve their performances almost linearly, until, after approximately 4% of the predictions have been discarded, the standard neural network slows heavily to improve the performance while the Bayesian model keeps on increasing until it reaches almost 100% of accuracy after discarding approximately 12% of the test images.

This proves that Bayesian models, not only regularize better their parameters, but are also able to isolate bad prediction, sacrificing only a small part of the good ones.

4.5 Uncertainty visualization

The main aim of this section is to understand the predictive uncertainty analyzed so far by looking at the images corresponding to the most relevant predictions, hence exploiting the choice of a comprehensible dataset such as MNIST. This process further underlines the added capability to perform data analysis using Bayesian models.

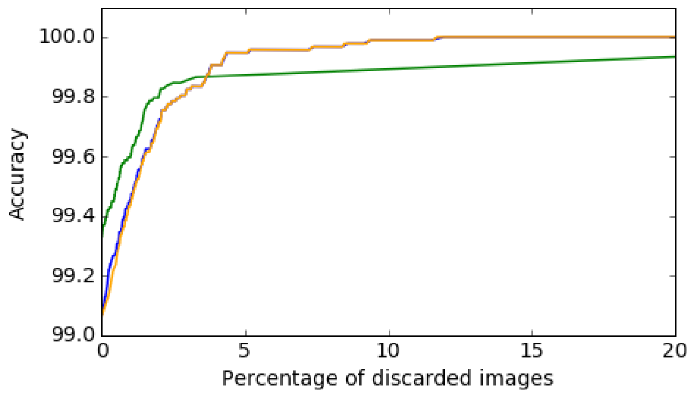


Fig. 4.20 Accuracy versus corresponding percentage of discarded predictions by using the variance (orange) and output (blue) of a Bayesian neural network and the output of a standard neural network (green).

For each selected image, the corresponding output is displayed below in the form of a bar plot, where each bar correspond to a class, the predicted class is green (if correct) or red (if wrong) and the others are blue.

Figure 4.21 illustrates a sample of 20 images which have been predicted with a bad label but with very high uncertainty (variance ≥ 0.2). All this images are very badly written digits that are hard to interpret even for a human eye and in the worst cases do not even look like the corresponding label, e.g. the four that looks like a nine. The variance can also be used to find corrupted data, for example, transposed or truncated images.

The output corresponding to each of this images is never certain, the maximum value does not get over 0.7 and it often has more classes with similar values. Moreover, even if the prediction is wrong, it is possible to see that the correct label is often among the top three, in fact the top3 accuracy is 99.95%.

Figure 4.22 shows instead a sample of images with high variance that have been predicted correctly. These images look as bad as the previous ones in fact they share a very high predictive variance and the corresponding outputs look uncertain as well. The only difference lays in the fact that, for these images, the correct label was slightly higher that the wrong one instead of the opposite.

This couple of figures is fundamental to understand the expressive power of uncertainty modeling. The predictive variance can identify a great amount of wrong predictions and, furthermore, it can be used to isolate bad input data.

In order to cross check this statement, a sample of images with low uncertainty (predictive variance ≤ 0.02) is displayed in Figure 4.23. As we already knew from the variance plots above (Figures 4.17 and 4.18), the vast majority of images with low uncertainty are predicted

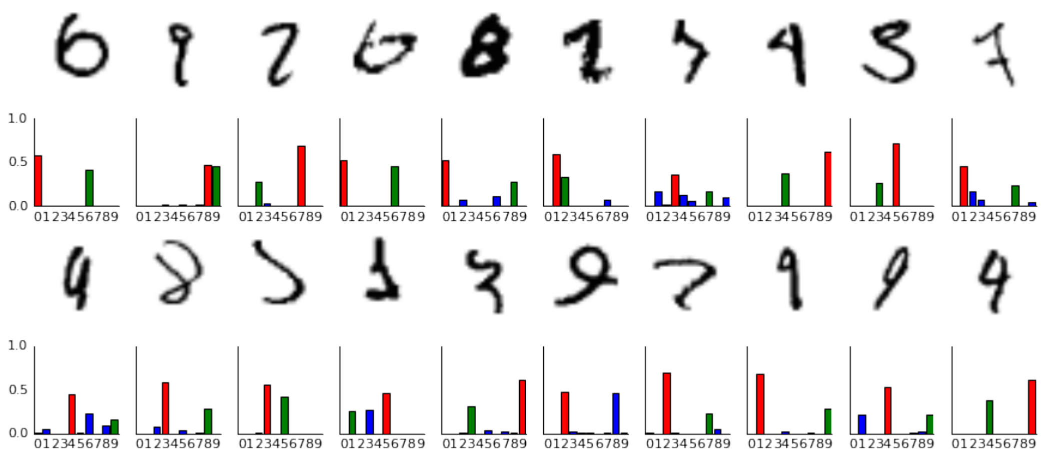


Fig. 4.21 Sample of 20 images and the corresponding output. The images have been predicted wrong by a Bayesian neural network but with high variance.

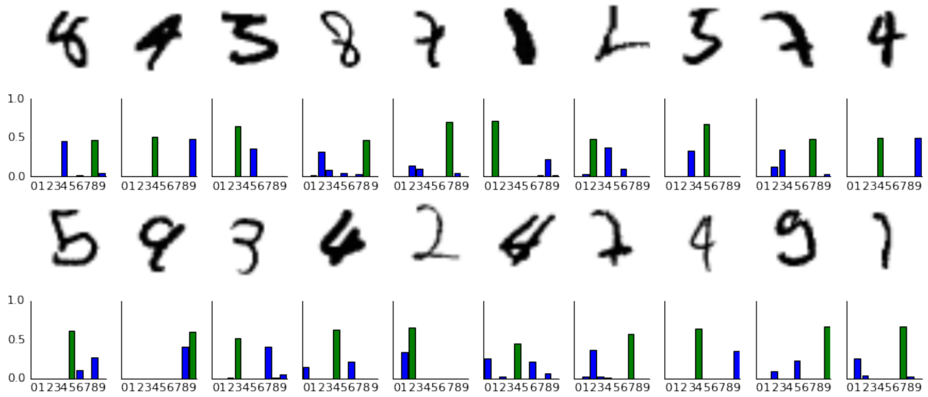


Fig. 4.22 Sample of 20 images and the corresponding output. The images have been predicted correctly by a Bayesian neural network but with high variance.

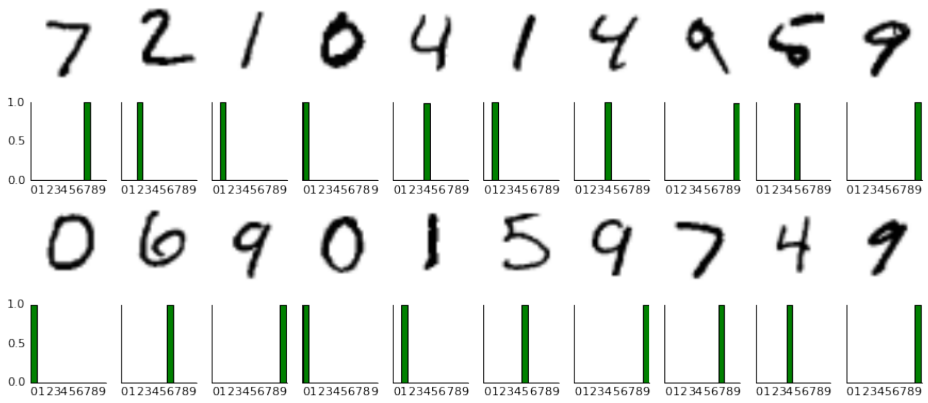


Fig. 4.23 Sample of 20 images and the corresponding output. The images have been predicted with low variance by a Bayesian neural network.

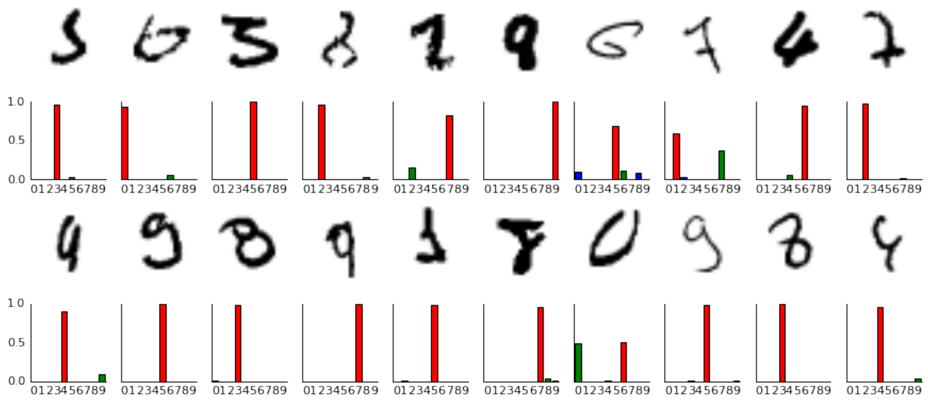


Fig. 4.24 Sample of 20 images and the corresponding output. The images have been predicted wrong by a standard neural network.

correctly, in fact, they all look very clean and the corresponding outputs contain a single predominant class with output value very close to one.

Finally, Figure 4.24 shows a sample of images which have been badly predicted by the standard neural network. Even if this model has a great accuracy, it is not really able to discriminate between certain and uncertain predictions, as well as between good and bad input data. The wrongly predicted images look exactly like the images with very high predictive variance but the corresponding output presents a single spike corresponding to the wrong class, leading to think that the label can be assigned with high probability.

Because of the strong correlation between predictive variance and output value of the Bayesian neural network, the two modelling approaches considered so far can be compared by using the corresponding outputs. Figure 4.25 is a scatter plot of the output maximum values, point estimate and Bayesian, for each test image. This plot clearly shows the main difference between the two models, in the same way as showed in the previous figures, the standard neural network tends to output mostly values close to one while the Bayesian model allows a higher number of smaller values.

Figure 4.26 shows images from the bottom right corner and the top left one of Figure 4.25. The left ones are images of digits that are clearly shady, ambiguous or bad written. These images are highlighted by the Bayesian model with a high uncertainty and therefore

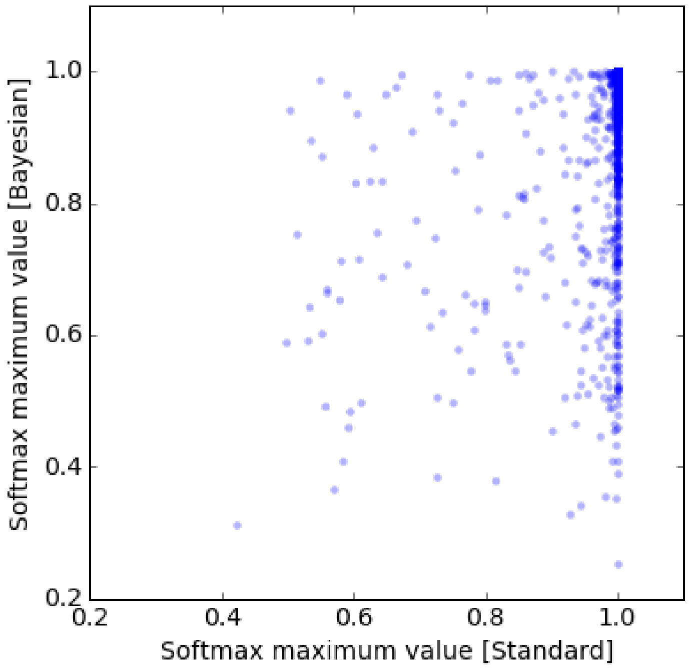


Fig. 4.25 Scatter plot where each point corresponds to a test image and its coordinates are the maximum values of the output of a normal and a Bayesian neural network.

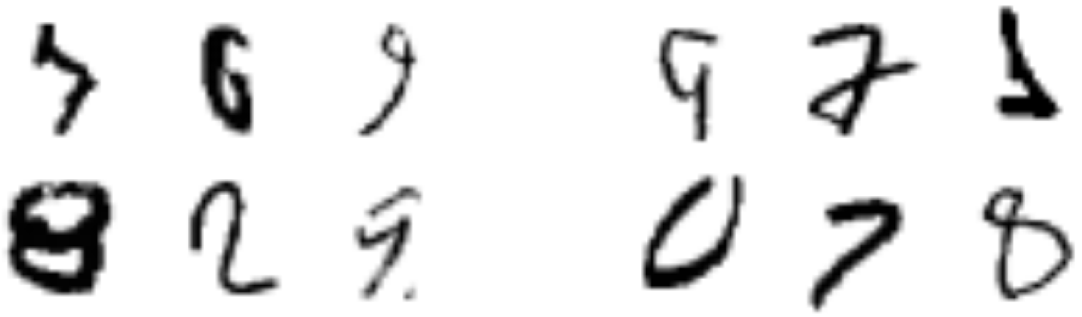


Fig. 4.26 Images with low output from the Bayesian neural network but high output from the standard one (left), and low output from the standard neural network but high output from the Bayesian one (right).

a low output value, while the normal neural network is able to predict them with a higher accuracy but with a very high output value as well, meaning that it makes no difference between these and other well written digits.

The second sample of images represents instead the opposite case, where the Bayesian model is very certain while the other one has a low output value. Even if these images do not look perfect they are not comparable to the other ones because they look much cleaner and it is easy to understand the corresponding label.

4.6 Out-of-distribution samples

In the previous section we visualized the images corresponding to the highest and lowest predictive uncertainty and we compared the Bayesian output to the output of a standard convolutional neural network.

The main advantage of the Bayesian model lays in its capacity to recognize bad input data and therefore improve the accuracy by discarding them. The same property can be used by the model to isolate out-of-distribution data, meaning images that are different from the training data not in terms of quality but in terms of context, they don't correspond to any label used by the model.

In order to evaluate the ability of this model to discriminate these images we used two dataset of images using the same format as MNIST (same dimension and number of channels) but contextually different. Figure 4.27 and Figure 4.28 respectively show samples from the EMNIST and notMNIST datasets. The first one (Extended MNIST) is an extension of the MNIST dataset containing more images of digits as well as uppercase and lowercase letters.

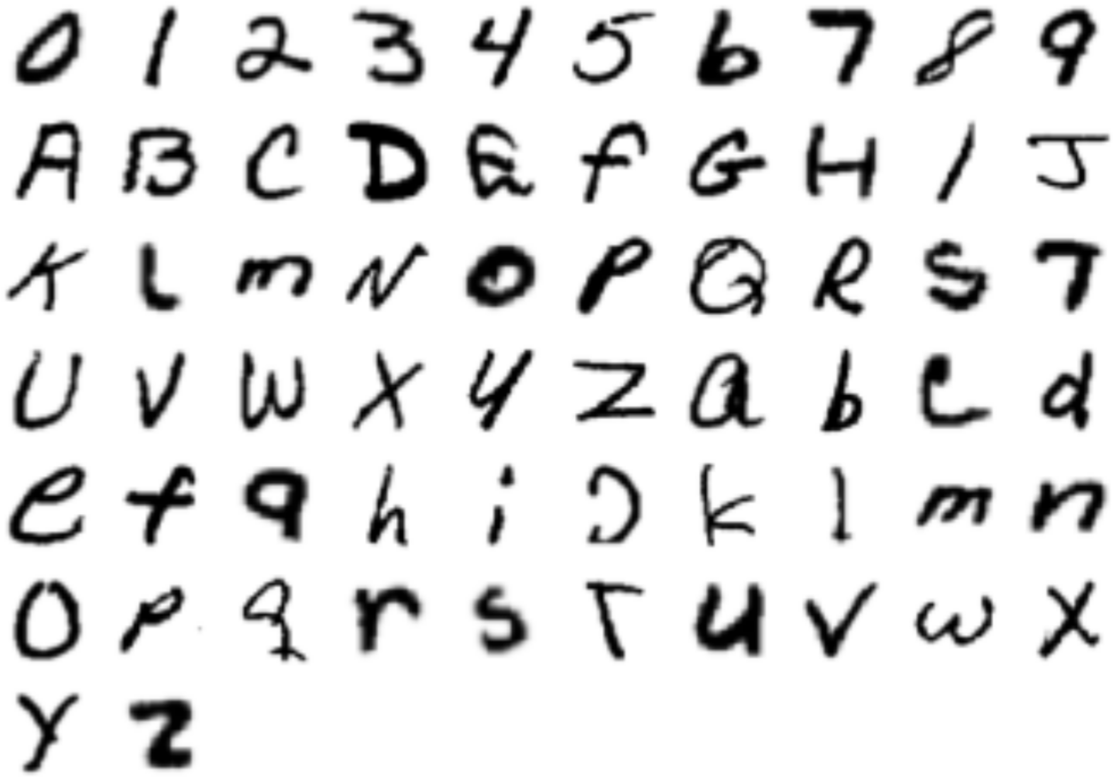


Fig. 4.27 Sample of images illustrating each class of the EMNIST dataset.

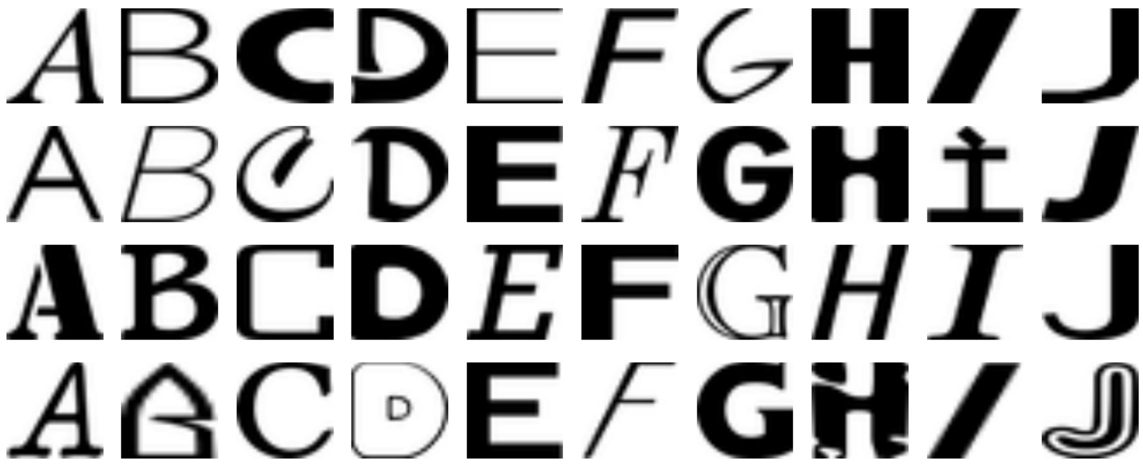


Fig. 4.28 Different images for each class of the notMNIST dataset.

The notMNIST dataset instead has again only 10 classes corresponding to images of the letters from A to J illustrated with different fonts.

4.6.1 EMNIST dataset

In order to analyse this dataset we initially fed all the 116323 images to the model and looked at the distribution of the predictive variance showed in Figure 4.29 (left). This figure does not look as expected, in fact, since the dataset contains a lot of letters that do not look like digits, we expected a very high spike on high variances instead of low ones. By looking at the images with lower variance we realized that most of them correspond to digits, that, on the other side, we expect to have a very low predictive variance.

After removing the images of numbers digits, the dataset was left with only 58405 images. The corresponding predictive variance distribution is displayed in Figure 4.29 (right) where we can see that the lower variance spike decreased from a count of more than 50000 predictions to less than 15000. Even if this decrease is notable, the variance distribution still has a big spike on the lower uncertainties, therefore we proceeded by analyzing the predictive variance by looking at each class of the dataset.

Figure 4.30 illustrates predictive variance, predicted classes and images for some classes of EMNIST with either low (first two rows) or high (last two rows) predictive variance. The most certain predictions correspond to letters that look very similar to numerical digits and this can be enforced by the count of the predicted classes. Most of the ’O’s and ’I’s have very low uncertainty and they are mostly predicted as zeros and ones respectively.

A similar behaviour appears for classes which have higher predictive variance, the model is able to separate the images that look more like numbers from the ones that don’t. For

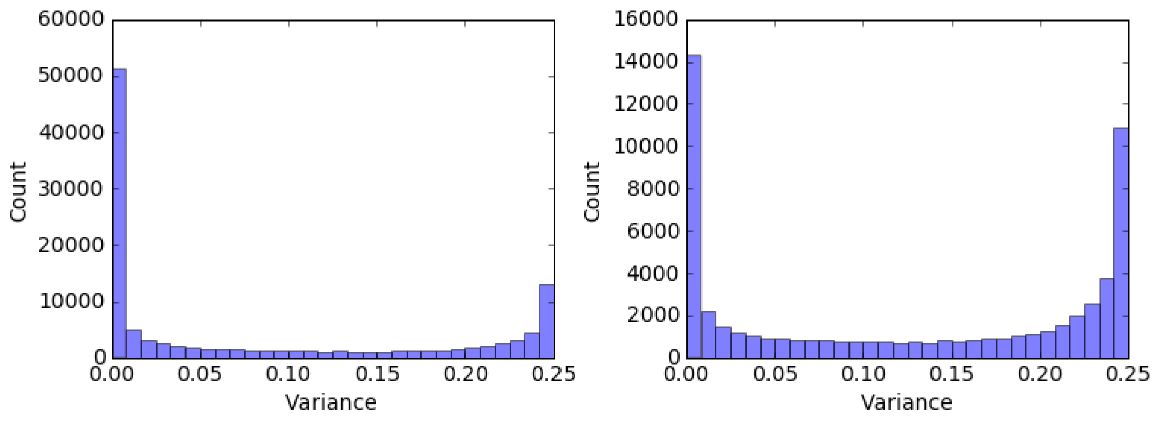


Fig. 4.29 Histogram of the predictive variance of all the images of the EMNIST dataset (left) and only the images containing letters (right).

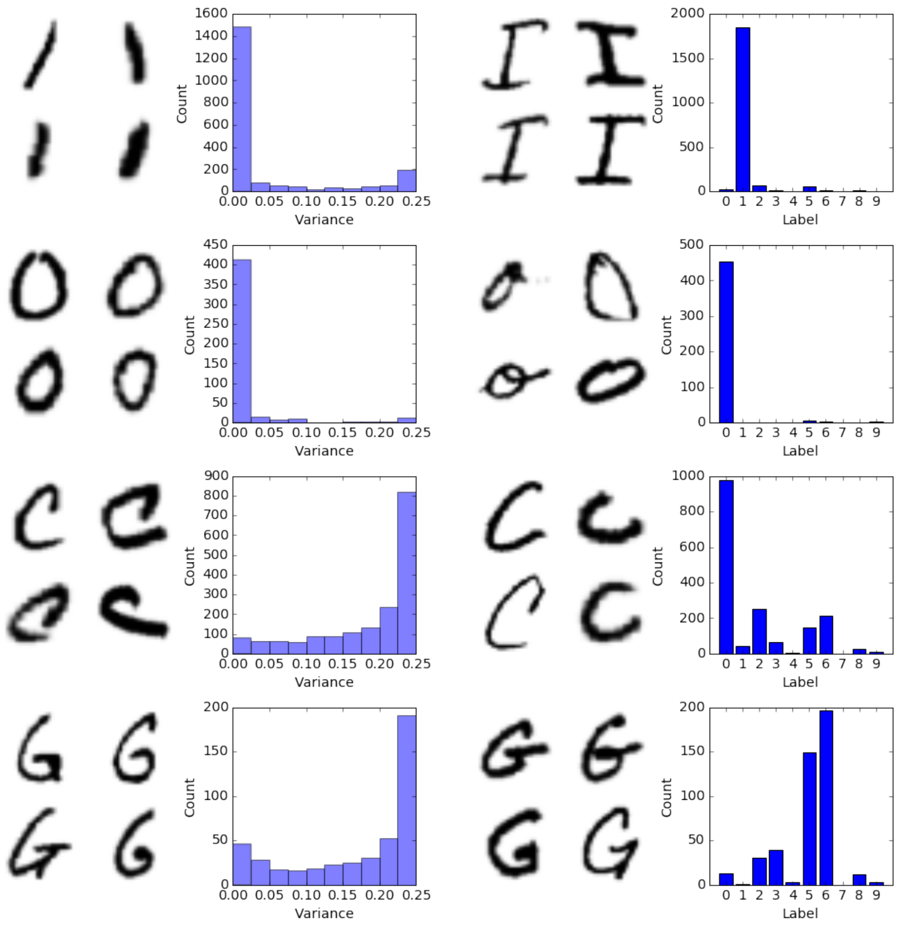


Fig. 4.30 Images of 'I', 'o', 'C' and 'G' from the EMNIST dataset. The leftmost column illustrates 4 samples predicted with low variance while the third column displays 4 samples predicted with high variance.The second column shows the histogram of the predictive variance of the class and the rightmost column contains the barplots with the counts of the predictions corresponding to the MNIST classes.

example, the 'C's with lower variance look more like zeros than the ones with higher variance, as well as 'G's with low uncertainty look more like a six.

The complete analysis of the predictive variance for each class of the EMNIST dataset is illustrated in Appendix 6.

4.6.2 notMNIST dataset

Images from the notMNIST dataset have been analysed in the same way as the EMNIST ones. Figure 4.31 shows the predictive variance distribution for the 18724 images of this dataset while Figure 4.33 shows the per class variance analysis as in the previous figures. Since this dataset is very different from the original MNIST dataset, the uncertainty is much

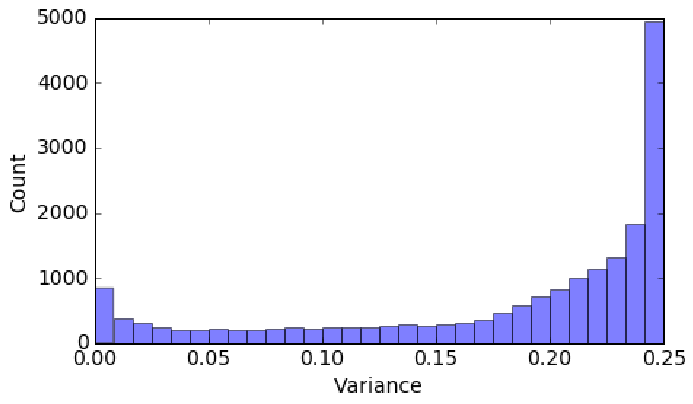


Fig. 4.31 Histogram of the predictive variance of all the images of the notMNIST dataset.

higher because the model easily recognizes images not belonging to the original domain of MNIST images. The only images being predicted with low uncertainty are some ’I’s and ’J’s looking like ones and some ’A’s and ’H’s which also on the EMNIST dataset presented the same behaviour because they have the same features as a four, i.e. they contain corners.

Finally, Figure 4.32 shows the softmax maximum value for all the images of the dataset. While the standard neural network behaves as usual and shows very high outputs for most of the images, the Bayesian model outputs lower values, signaling that the predictions must be uncertain.

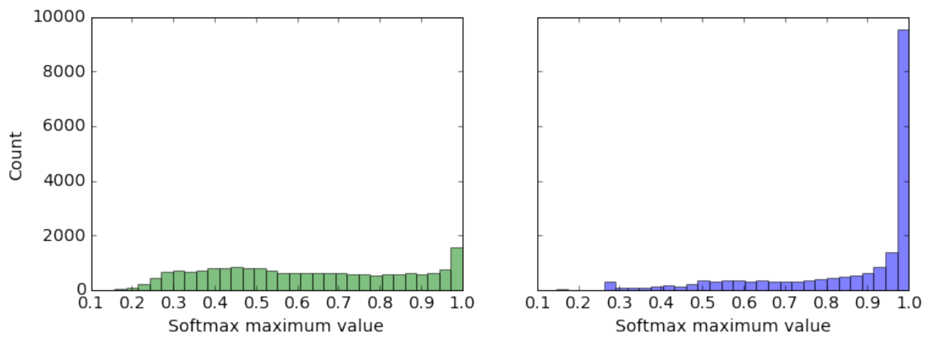


Fig. 4.32 Histogram of the output maximum value of all the images of the notMNIST dataset using a Bayesian neural network (left) and a standard neural network (right) trained on the MNIST dataset.

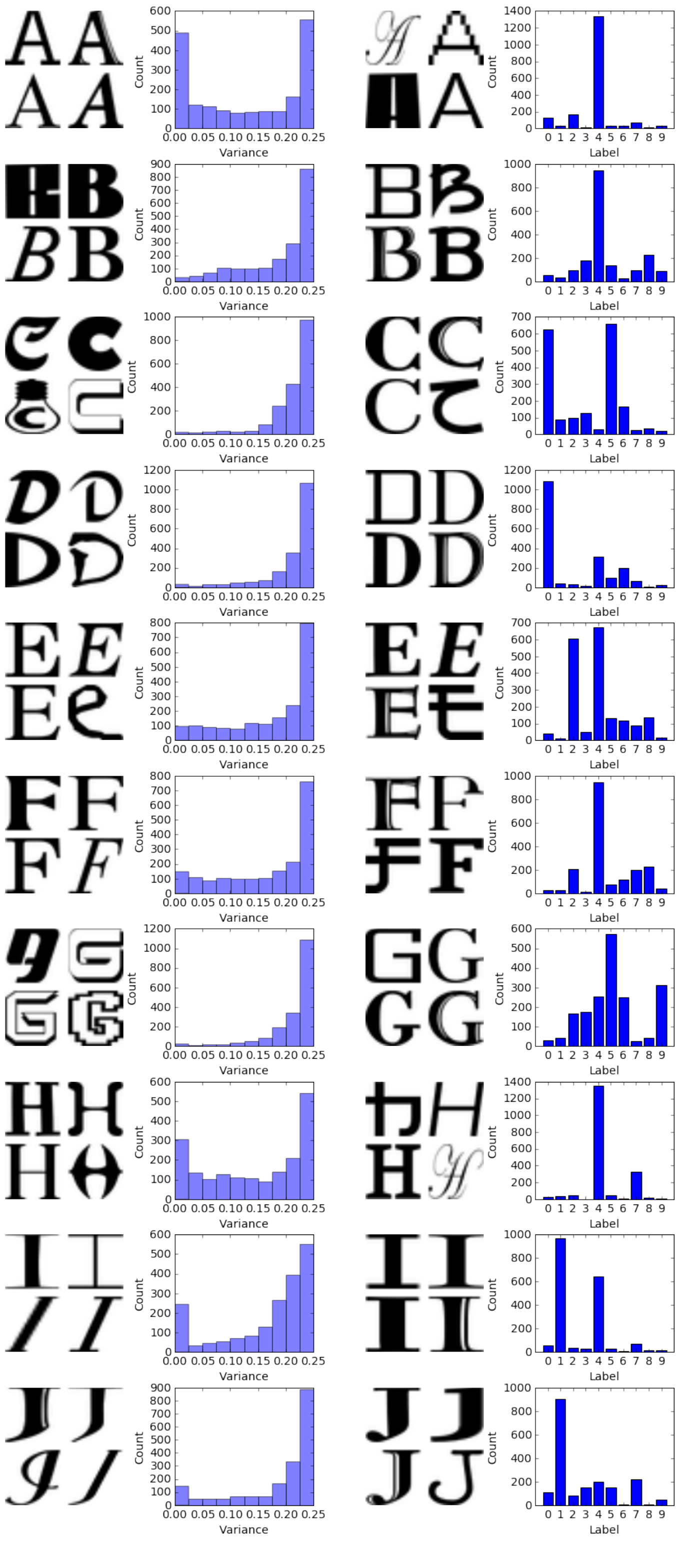


Fig. 4.33 For each class of the notMNIST dataset: images predicted with low variance (leftmost column) and with high variance (third column), histogram of the predictive variance (second column) and barplot containing the counts of the predictions corresponding to the MNIST classes.

Chapter 5

Biomedical use cases

5.1 Introduction

Biological and medical technologies have been exploited in the latest years in the same way as in many other fields to provide huge volumes of data. Biomedical data are generally complex, high-dimensional, heterogeneous and unstructured therefore exhibit some of the most interesting challenges for modern deep learning approaches. Such methods have been proved to be well performing on a vast amount of tasks and are now being tested on biomedical applications such as protein structure classification and prediction, medical image classification or genomic sequence analysis.

The main advantage of deep neural networks over more traditional machine learning approaches lays in their ability to learn the features directly from the data following an automated optimization process. Such features can then be exploited to perform different tasks spanning from prediction to clustering (Cao et al., 2018; Miotto et al., 2017; Wainberg et al., 2018).

Gaining knowledge from biomedical data presents its own problematic challenges, mainly related to the high noise in the observations as well as the small number of samples and annotations due to very expensive time and money constraints.

In this section we analyse the application of Bayesian convolutional neural networks to images coming from high content screenings as well as high throughput screenings. Such datasets present clearly some of the previously mentioned issues, they have a low number of observations corrresponding to a high number of potential classes and they are hard to label correctly even for a human expert in the field. Moreover, a certain phenotype may be present only in a portion of the cells belonging to a certain image and the noise in the image acquisition process may provide the scientist with a small number of very different replicates for the same experiment.

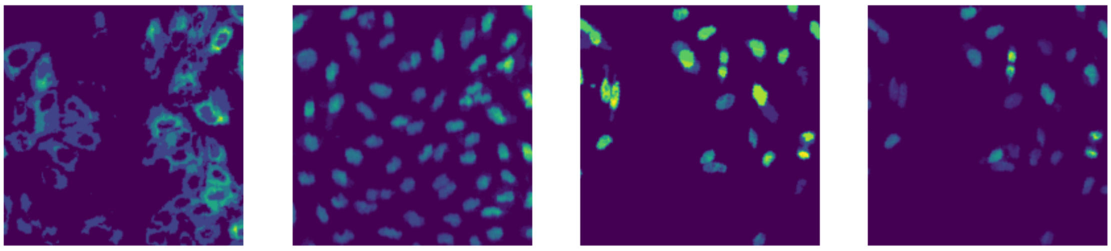
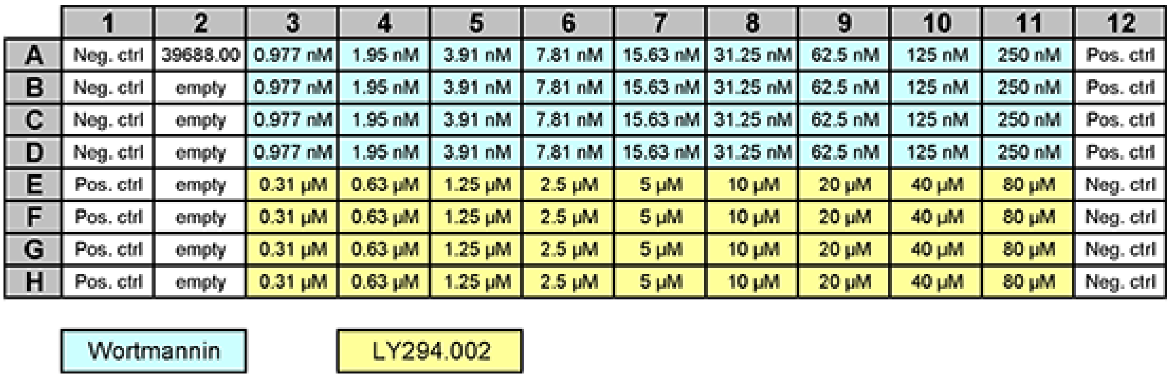


Fig. 5.1 Channels of an image sample of the negative control (left) and positive control (right) of the BBBC013 dataset.

5.2 BBBC013 dataset

We conducted the first experiment on the BBBC013 dataset provided by Ilya Ravkin, available from the Broad Bioimage Benchmark Collection (Ljosa et al., 2012). The BBBC013 dataset is made of images coming from a 96-well plate. This images contain the cytoplasm to nucleus translocation of the Forkhead (FKHR-EGFP) fusion protein in stably transfected human osteosarcoma cells, U2OS. In proliferating cells, FKHR is localized in the cytoplasm. Even without stimulation, Forkhead is constantly moving into the nucleus, but is transported out again by export proteins. Upon inhibition of nuclear export, FKHR accumulates in the nucleus. In this assay, export is inhibited by blocking PI3 kinase / PKB signaling by incubating cells for 1 h with Wortmannin or with the compound LY294002 as illustrated in Figure 5.1. Both drugs are considered positive controls in the assay. Nuclei are stained with DRAQ, a DNA stain. The image size is 640 x 640 pixels and they have two channels, one for the FKHR-GFP and another for the DNA. The images provided are 4 replicas of the 9-point dose curve for each of the two positive control drugs (Wortmannin and LY294002). The



	1	2	3	4	5	6	7	8	9	10	11	12
A	Neg. ctrl	39688.00	0.977 nM	1.95 nM	3.91 nM	7.81 nM	15.63 nM	31.25 nM	62.5 nM	125 nM	250 nM	Pos. ctrl
B	Neg. ctrl	empty	0.977 nM	1.95 nM	3.91 nM	7.81 nM	15.63 nM	31.25 nM	62.5 nM	125 nM	250 nM	Pos. ctrl
C	Neg. ctrl	empty	0.977 nM	1.95 nM	3.91 nM	7.81 nM	15.63 nM	31.25 nM	62.5 nM	125 nM	250 nM	Pos. ctrl
D	Neg. ctrl	empty	0.977 nM	1.95 nM	3.91 nM	7.81 nM	15.63 nM	31.25 nM	62.5 nM	125 nM	250 nM	Pos. ctrl
E	Pos. ctrl	empty	0.31 µM	0.63 µM	1.25 µM	2.5 µM	5 µM	10 µM	20 µM	40 µM	80 µM	Neg. ctrl
F	Pos. ctrl	empty	0.31 µM	0.63 µM	1.25 µM	2.5 µM	5 µM	10 µM	20 µM	40 µM	80 µM	Neg. ctrl
G	Pos. ctrl	empty	0.31 µM	0.63 µM	1.25 µM	2.5 µM	5 µM	10 µM	20 µM	40 µM	80 µM	Neg. ctrl
H	Pos. ctrl	empty	0.31 µM	0.63 µM	1.25 µM	2.5 µM	5 µM	10 µM	20 µM	40 µM	80 µM	Neg. ctrl

Wortmannin LY294.002

Fig. 5.2 Platemap of the BBBC013 dataset from the corresponding webpage: https://data.broadinstitute.org/bbbc/BBBC013/.

platemap illustrated in Figure 5.2 is a standard 96-well plate, containing 9 points of 2-fold dilutions (columns 3-11) of either Wortmannin or LY294002. Negative and positive controls have no drug and 150 nM Wortmannin added, respectively.

Previous work has already been done on this dataset in order to compute the Z'-factor and the V-factor of both the compounds used for the positive control (Carpenter et al., 2006; Logan and Carpenter, 2010). An interesting analysis, exploiting deep neural networks, has been performed by Godinez et al. (2017).

In the latter publication, the authors train a deep neural network to predict the compound activation, that is if the positive control or the negative one is used. In order to do so they only use the images corresponding to the negative and positive control, without considering the intermediate concentrations. Furthermore, they predict all the images in the dataset and they take the median of the probabilities computed for images corresponding to replicates of a given concentration. Finally, they plot these median values as a function of the concentration. These plots show that the phenotype probability quantitatively correlates with the treatment concentrations, in fact the probability is closer to 0 for smaller concentrations and to 1 for greater ones.

We decided to reproduce this experiment and study the probability of the phenotype for the concentration that is closer to the EC_{50} value, that is the concentration of a drug that gives half-maximal response.

Since the training set is very small, in fact only 16 of the 88 available images are used, and the phenotype difference is very clear, we decided to use a very simple architecture

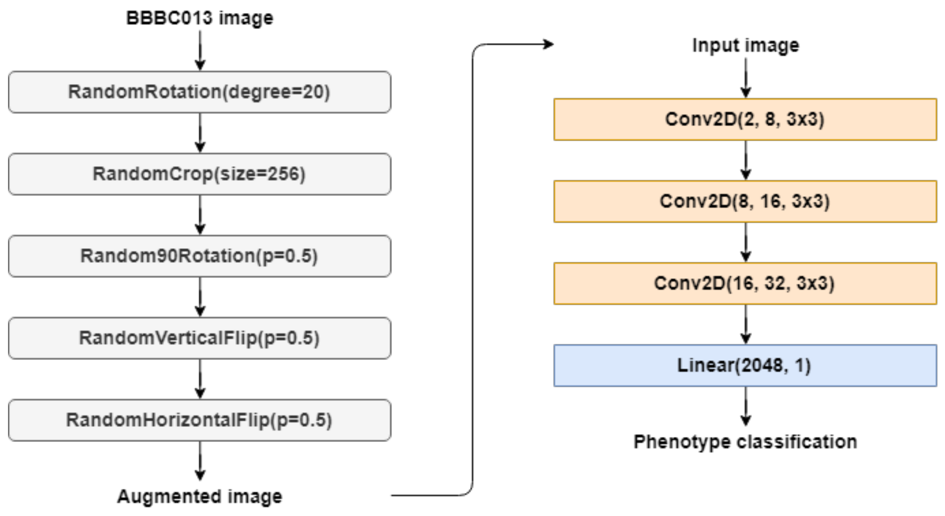


Fig. 5.3 BBBC013 data augmentation pipeline (left) and neural network architecture used to perform classification (right).

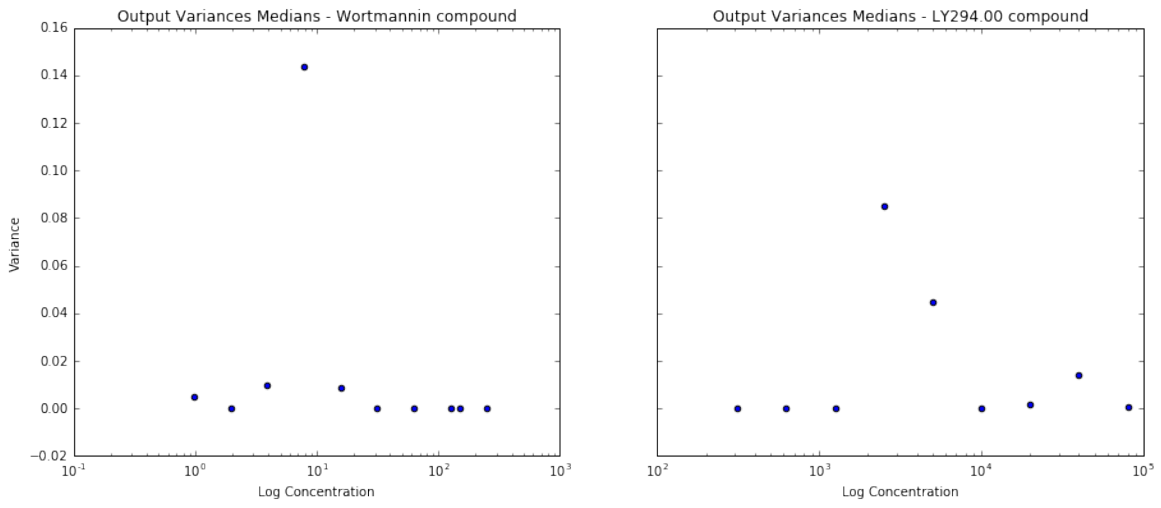


Fig. 5.4 Median of the predictive variance of the positive control compounds for each concentration.

consisting of three convolutional layers with kernel size equal to 3 and one fully connected layer.

In order to be able to fully exploit the few available images we performed some augmentation steps before the training and evaluation. We initially rotated each image by a random angle between -20 and 20 degrees. Then we sampled patches of size 256 x 256. This was possible because the cell distribution and the phenotype have a uniform spatial distribution in the images. This enforces the idea that it is relatively simple to perform inference on such dataset and therefore a simple architecture is able to reach good performances. The ultimate step of data pre-processing includes further random rotations and flips. The data augmentation pipeline and architecture used with the BBBC013 dataset are illustrated in Figure 5.3. The final dataset is augmented by a factor of 500 with respect to the original size.

Finally, Figure 5.4 illustrates the median of the predictive variance for each concentration of the compounds used for the positive control. The maximum value of the uncertainty is the one corresponding to the concentration that is the closest to the EC_{50} value because it is the one corresponding to the most ambiguous images.

5.3 BBBC021 dataset

The most important experiment we performed, required the use of the image set BBBC021v1 (Caie et al., 2010), available from the Broad Bioimage Benchmark Collection (Ljosa et al., 2012).

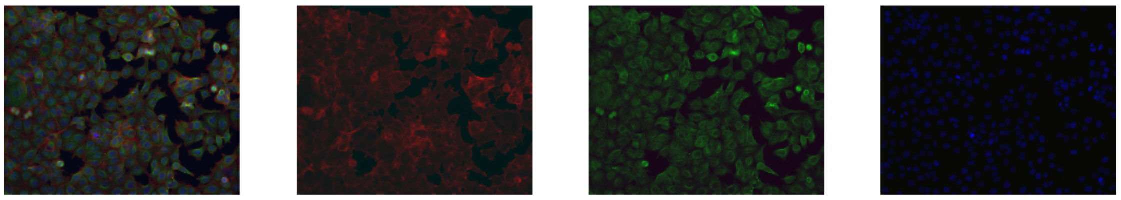


Fig. 5.5 From left to right: image sample from the BBBC021 dataset corresponding to the negative control (DMSO) and its three channels, F-actin, B-tubulin and DNA.

The aim of the experiment is to perform phenotypic profiling, that is to summarize multiparametric, feature-based analysis of cellular phenotypes of each sample so that similarities between profiles reflect similarities between samples. Profiling is well established for biological readouts such as transcript expression and proteomics. Image-based profiling, however, is still an emerging technology.

This image set provides a basis for testing image-based profiling methods with respect to their ability to predict the mechanisms of action (MOA) of a compendium of drugs. The mechanism of action refers to the specific biochemical interaction through which a drug substance produces its pharmacological effect. The dataset uses a physiologically relevant p53-wildtype breast-cancer model system (MCF-7) and a mechanistically distinct set of targeted and cancer-relevant cytotoxic compounds that induces a broad range of gross and subtle phenotypes. The cancer cells have been treated for 24 h with a collection of 113 small molecules at eight concentrations.

The dataset is composed by 13,200 fields of view of size 1024 x 1280 imaged in three fluorescent channels corresponding to DNA, F-actin, and B-tubulin as illustrated in Figure 5.5.

A subset of the compound-concentrations have been identified as clearly having one of 12 different primary mechanims of action. Mechanistic classes were selected so as to represent a wide cross-section of cellular morphological phenotypes. The differences between phenotypes were in some cases very subtle: only 6 of the 12 mechanisms were identified visually, the remainder were defined based on the literature.

Previous work has been done on this dataset to perform MOA classification of the known phenotypes using different approaches (Ando et al., 2017; Ljosa et al., 2013; Pawlowski et al., 2016) and again the most relevant results are those exploiting deep convolutional neural networks (Godinez et al., 2017). In the latter publication, the authors use a multi scale convolutional neural network that is composed of 7 convolutional neural networks processing the images at different scales as illustrated in Figure 5.6.

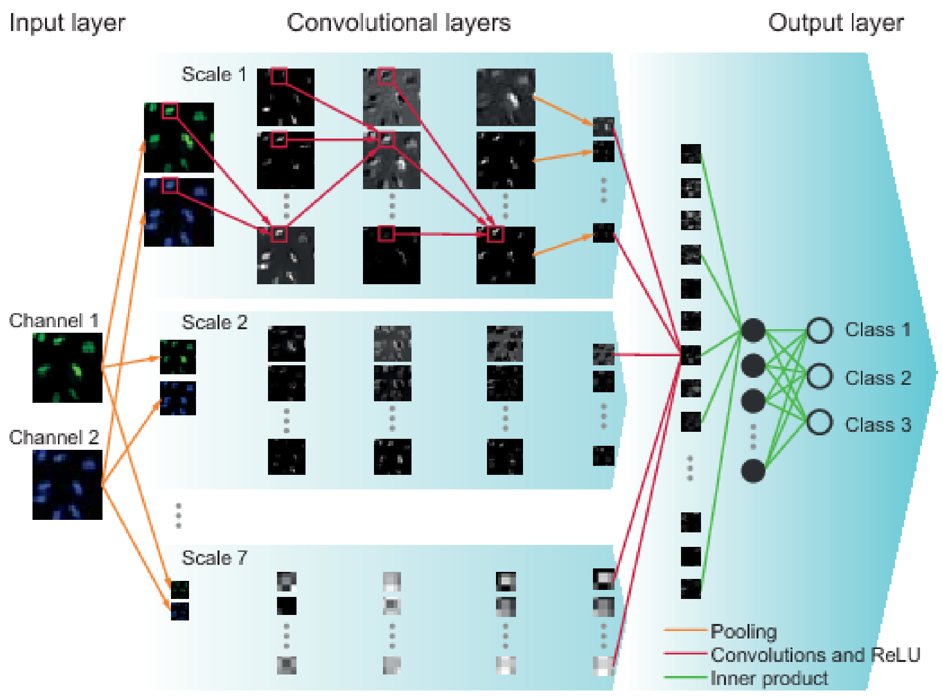


Fig. 5.6 Architecture of the multi scale convolutional neural network used by Godinez et al. (2017).

We decided to follow up on this work and test the performance of a Bayesian multi scale convolutional neural network in classifying known MOAs as well as identifying out-of-distribution samples, that is MOAs that are not included in the training data.

We decided to modify the initial multi scale architecture presented by Godinez et al. (2017) because it was overparameterized. In fact, by introducing batch normalization layers and decreasing the number of fully connected layers performing the final classification after the convolutional neural networks, the performance did not decrease and the training became faster.

In order to deal with the highly unbalanced classes of BBBC021 (see Table 5.1) we performed two steps. Initially we truncated the DMSO and Microtubule stabilizers classes to the median length of the classes. 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.

In order to evaluate the accuracy of MOA classification we followed a Leave-One-Compound-Out (LOCO) cross-validation approach. In order to do so we trained the neural

Mechanism of Action	**Compound**	**Concentrations** $[\mu M]$	**N**
Actin disruptors	cytochalasin B	[10. 30.]	24
Actin disruptors	cytochalasin D	[0.3]	12
Actin disruptors	latrunculin B	[1. 3.]	24
Aurora kinase inhibitors	AZ-A	[0.1 0.3 1. 3. 10. 30.]	72
Aurora kinase inhibitors	AZ258	[0.1 0.3 1.]	36
Aurora kinase inhibitors	AZ841	[0.1 0.3 1.]	36
Cholesterol-lowering	mevinolin/lovastatin	[1.5 5. 15.]	36
Cholesterol-lowering	simvastatin	[2. 6. 20.]	36
DNA damage	chlorambucil	[10.]	12
DNA damage	cisplatin	[10.]	12
DNA damage	etoposide	[1. 3. 10.]	36
DNA damage	mitomycin C	[0.1 0.3 1. 3.]	48
DNA replication	camptothecin	[0.003 0.01 0.03]	36
DNA replication	floxuridine	[10. 30.]	24
DNA replication	methotrexate	[10.]	12
DNA replication	mitoxantrone	[0.003 0.01]	24
Eg5 inhibitors	AZ-C	[0.001 0.003 0.01 0.03 0.1 0.3 1.]	84
Eg5 inhibitors	AZ138	[0.03 0.1 0.3 1. 3.]	60
Epithelial	AZ-J	[1. 3. 10.]	36
Epithelial	AZ-U	[1. 3. 10.]	36
Epithelial	PP-2	[3. 10.]	16
DMSO	DMSO	[0.]	96
Kinase inhibitors	PD-169316	[3. 10.]	16
Kinase inhibitors	alsterpaullone	[1. 3.]	16
Kinase inhibitors	bryostatin	[0.3]	8
Microtubule destabilizers	colchicine	[0.03]	12
Microtubule destabilizers	demecolcine	[0.3 1. 3. 10.]	48
Microtubule destabilizers	nocodazole	[1. 3.]	24
Microtubule destabilizers	vincristine	[0.003 0.01 0.03 0.1 0.3 1. 3.]	84
Microtubule stabilizers	docetaxel	[0.03 0.1 0.3]	36
Microtubule stabilizers	epothilone B	[0.1 0.3 1.]	36
Microtubule stabilizers	taxol	[0.3]	24
Protein degradation	ALLN	[3. 100.]	24
Protein degradation	MG-132	[0.1 3.]	24
Protein degradation	lactacystin	[10.]	12
Protein degradation	proteasome inhibitor I	[0.1 3.]	24
Protein synthesis	anisomycin	[0.3 1.]	24
Protein synthesis	cyclohexamide	[5. 15. 50.]	36
Protein synthesis	emetine	[0.1 0.3 1.]	36

Table 5.1 Metadata and number of samples (N) of the annotated subset of BBBC021.

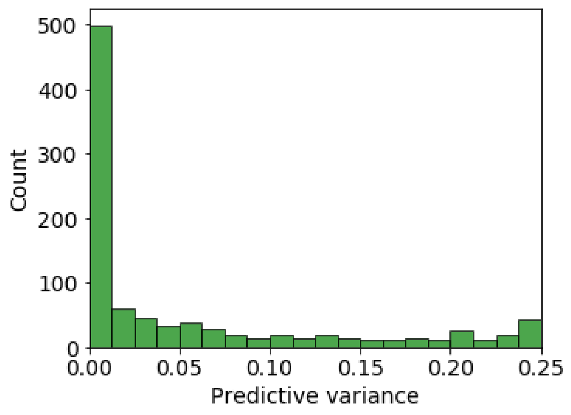


Fig. 5.7 Histograms of the predictive variance of the correct (left) and wrong (right) predictions of the BBBC021 images using a Bayesian multi scale neural network with a Leave-One-Compound-Out validation scheme.

network on all the compounds except for one, tested on the held out compound and iterated over all the compounds.

After gathering together the predictions for each compound when held out of the training data, we were able to analyze the predictive variance distribution as previously done. Figure 5.7 illustrates the histograms of the predictive variance of all the images in the truncated dataset. The correct predictions are mainly performed with low uncertainty as can be seen on the left plot, while the histogram of the wrong predictions presents two main modes: one for wrong predictions with low uncertainty and another one for wrong predictions with high uncertainty.

As already explained in the previous chapters, we expect the wrong predictions to be made with very high uncertainty and to correspond to either corrupted input data or new classes that are not present in the training data. In this case, we know the images belong to the original domain of the training data therefore we need to investigate the presence of wrong predictions with low uncertainty as well as the quality of the input data leading to wrong predictions with high uncertainty. In order to proceed with such analysis we studied the predictive variance for each individual compound and analyzed the corresponding image samples.

While not all the compounds had perfect predictions or discriminating uncertainty due to the high amount of noise in the images and the low number of samples, most of them showed meaningful patterns.

Two (AZ-J and AZ-U) of the three compounds belonging to the epithelial class are entirely predicted correctly and mostly with low predictive variance. The most uncertain predictions correspond to images that are either empty, meaning that they have a very low

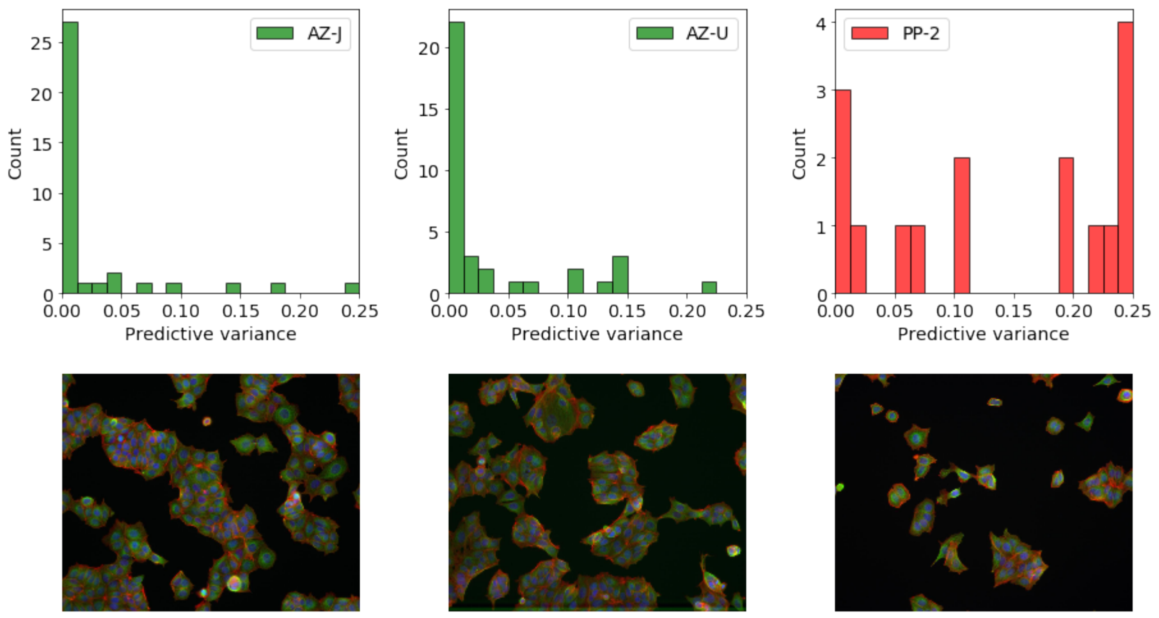


Fig. 5.8 Predictive variance of the epithelial compounds (top) and image samples (bottom).

cell density, or faded, out of focus. Figure 5.8 shows the predictive variance of the three compounds and the corresponding image samples. Images containing the last compound (PP-2) are entirely predicted wrong. These images shows slightly, or not at all, the corresponding phenotype, in fact the cells are not clustered together and, when they are, the clusters borders are not as smooth as they are for the other two compounds.

The two compounds (AZ-C and AZ138) labeled as Eg5 inhibitors have very low predictive variance and they have almost always been predicted correctly for all the different concentrations. The only wrong prediction, belonging to AZ138, has been highlighted with high uncertainty and predicted as microtubule stabilizer. By looking at the image it was possible to see that the two MOAs, the correct one and the prediction, look very similar, in fact the prediction score was 0.68 for microtubule stabilizers and 0.32 for Eg5 inhibitors, showing that the model knows about the potential correct prediction.

The three compounds (Anisomycin, Cyclohexamide and Emetine) labeled as protein synthesis have very low predictive variance and they have been predicted correctly in most of the cases. The only 3 images predicted wrongly all belong to Anisomycin. Figure 5.9 illustrates the distribution of the predictive variance of this compound and image samples from both the correct and wrong predictions. The predictions follow the same pattern illustrated in the previous chapter: correct predictions are made with low uncertainty while wrong ones have maximum predictive variance. By looking at the corresponding images it is possible to see how the correct predictions clearly show the protein synthesis phenotype while

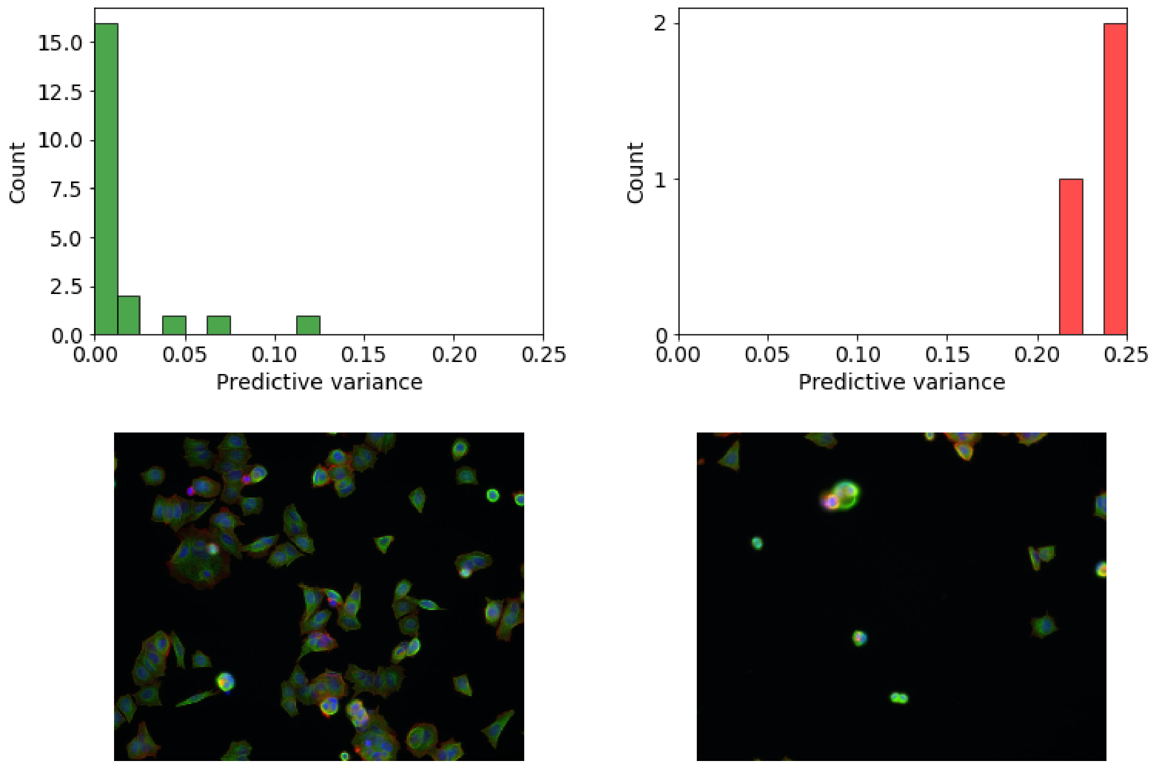


Fig. 5.9 Histograms of the predictive variance of correct (top-left) and wrong (top-right) predictions of the anisomycin compound and corresponding image samples (bottom).

the wrong ones have a very low cell density, the plate is almost entirely empty, and therefore can be classified as corrupted input data that may be discarded. Cyclohexamide presents one image predicted correctly but with high variance. The corresponding prediction scores are 0.72 for protein synthesis and 0.26 for epithelial, in fact, the cells of the corresponding image form some cluster like the ones classified as epithelial.

The DNA damage MOA does not present a very visual phenotype but the compounds belonging to this class are overall well performing, being predicted correctly and with low uncertainty.

Aurora kinase inhibitors compounds (AZ-A, AZ258 and AZ841) exhibit different predictive variances and both correct and wrong predictions. This MOA is not visually clear, in fact in some cases the images are classified as DMSO, that is the negative control used for the experiment.

The two compounds (mevinolin/lovastatin and simvastatin) classified as cholesterol lowering show both correct and wrong predictions but the overall predictive variance is meaningful because it is mainly low for the correct predictions and high for the wrong ones.

Mevinolin/lovastatin has 3 concentrations (1.5 μM, 5.0 μM and 15.0 μM). The lowest one is always predicted with high variance and mostly wrong because the phenotype is not very visual probably due to the fact that the concentration is too small. The middle concentration is always predicted correctly and the highest one is mainly predicted with higher variance both correctly and wrongly. The corresponding images show almost empty plates probably because the compound concentration is too high and it kills too many cells. Simvastatin has a less clean distinction between its different concentrations but the connection with the predictive variance is the same: the predictions are more uncertain when the MOA is not totally manifested or the plates are mostly empty.

The DNA replication phenotype is not very clear visually, it presents multiple nuclei very close to each other. The four corresponding compounds (camptothecin, floxuridine, methotrexate and mitoxantrone) exibit different predictive variances and both correct and wrong predictions. The predictive variance of floxuridine and methotrexate follows the usual pattern, it is low for good predictions and high for bad ones. While high variance predictions corresponding to floxuridine have low cell density, as already seen for other compounds, images containing methotrexate do not show any differences when being predicted with higher predictive variance. Camptothecin and mitoxantrone generally perform bad, there is no clear distinction between correct and wrong predictions with higher or lower predictive variance.

Actin disruptors have one of the clearest phenotypes, in fact actin is captured in one of the three fluorescent channels of the images and, for these compounds, it is fragmented around the cells. The cytochalasin compound is entirely predicted badly but, by looking at the corresponding images, it is possible to see that they are all out of focus probably due to some accident during the acquisition of the images. The corresponding predictive variance is not low but it is not very high as well, in fact it is more distributed across a range of neutral to high values. In the remaining two compounds (cytochalasin B and latrunculin B) the MOA can be clearly identified visually. Wrong predictions relative to cytochalasin B all have high predictive variance and they have low cell density even if they should have been predicted correctly because they still show the actin disruptors phenotype. Figure 5.10 shows the distribution of the predictive variance of latrunculin B and samples from the correct and wrong predictions. This compound has two concentrations (1.0 μM and 3.0 μM). The lower one clearly does not activate the corresponding MOA, in fact the corresponding images are mainly predicted wrongly and with high variance. The images with the highest concentration, instead, are predicted correctly and with low uncertainty because they clearly show the MOA. The only images preicted with higher uncertainty show artifacts due to issues in the image acquisition process.

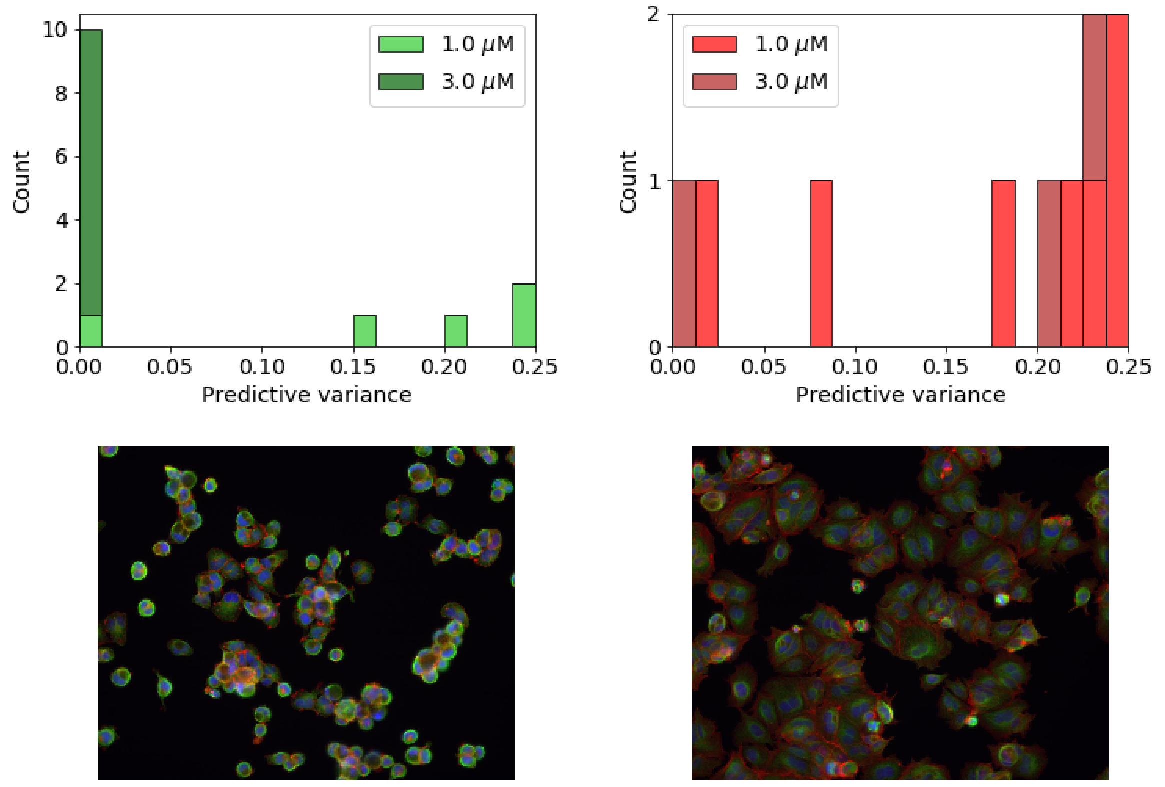


Fig. 5.10 Histograms of the predictive variance of correct (top-left) and wrong (top-right) predictions of the latrunculin B compound and corresponding image samples (bottom).

The compounds labeled as kinase inhibitors (PD-169318, alsterpaullone and bryostatin) and microtubule stabilizers (docetaxel, epothilone B and taxol) do not show a very clear phenotype but they are mainly predicted correctly and the predictive variance is low for correct predictions and high for the wrong ones.

Protein degradation is one of the worst performing MOAs, all the four compounds belonging to this class are predicted heterogeneously and they show different phenotypes. The corresponding predictive variance is neither discriminative nor informative.

Finally, microtubule destabilizers compounds (demelcocine, coclchicine, nocodazole and vincristine) are overall well performing, they are mostly predicted correctly and the predictive variance is meaningful. Figure 5.11 illustrates the comparison between demecolcine and colchicine. While the first compound is entirely predicted correctly and with low uncertainty, the latter is entirely predicted wrongly and with low uncertainty as well. By looking at the corresponding image samples it is clear that the concentration of colchicine is too low and it does not activate the MOA that has a very clear phenotype, the nuclei are fragmented into smaller circles. For this reason this compound has been predicted with the negative control

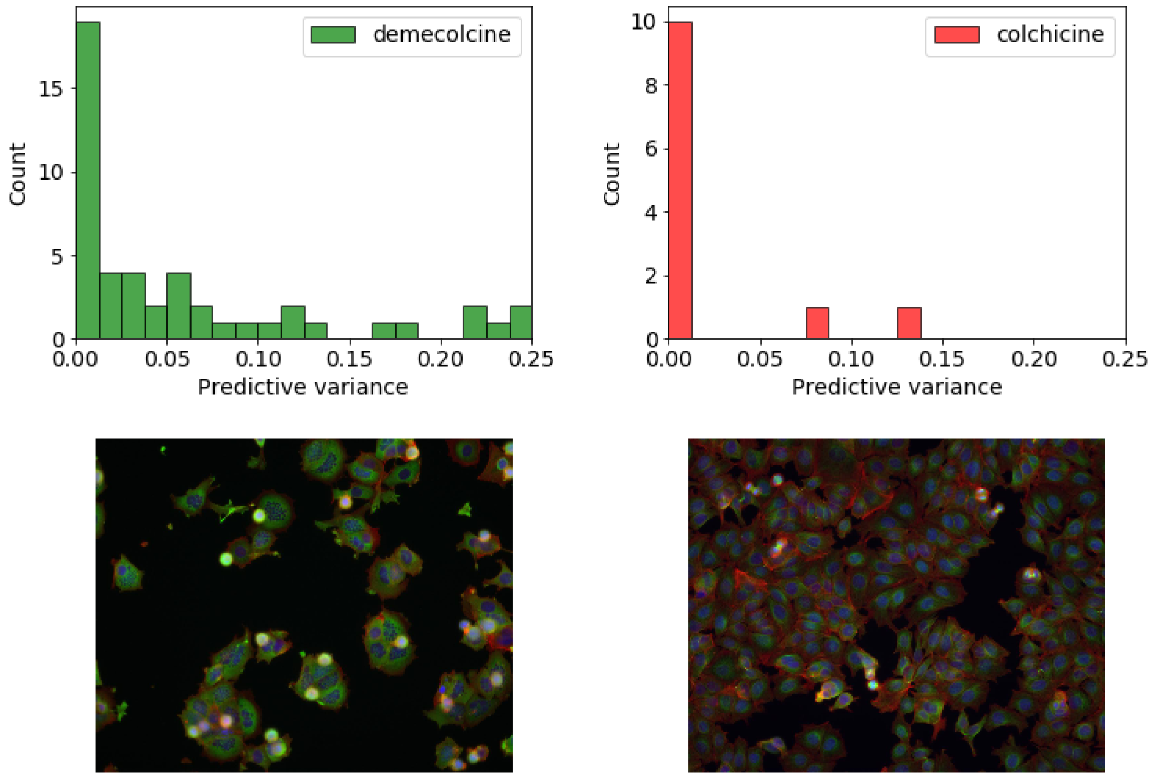


Fig. 5.11 Histograms of the predictive variance of two microtubule destabilizers compounds (top) and corresponding image samples (bottom).

class (DMSO). This can be considered as a correct prediction to the presence of wrongly labeled data.

Overall, the Leave-One-Compound-Out analysis showed some very interesting results. Not only the model was able to perform correct predictions with a high degree of accuracy, but also it was able to discriminate correct and wrong predictions using the predictive variance and to identify corrupted input data and wrongly labeled images.

Finally, in order to evaluate the ability of the neural network to identify new mechanisms of action we performed Leave-One-MOA-Out (LOMOAO) cross-validation. By doing so, every time the model is trained by leaving out an entire class that represents the out-of-distribution samples.

The predictive variance of LOMOAO validation is illustrated in Figure 5.12. It shows two main modes, one over lower uncertainty and one over higher uncertainty. This pattern is different than what we expected, that is to see all the predictions being made with high predictive variance. The reason for the predictions with lower predictive variance is due to different reasons: some of the compounds can be classified using multiple MOAS; as previously underlined, half of the MOAs were not identified visually, and the corresponding images look very similar to each other; there is a very low number of observations for each MOA, therefore the model is not able to learn the right features that distinguish one phenotype from the other. The full analysis of the LOMOAO predictive variance for each MOA is illustrated in Appendix 6.

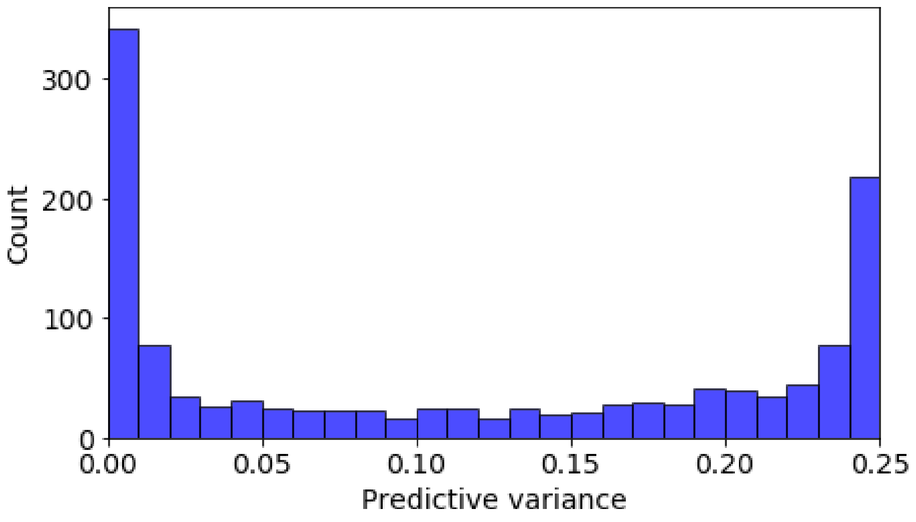


Fig. 5.12 Histograms of the predictive variance of the BBBC021 images using a Bayesian multi scale neural network with a Leave-One-MOA-Out validation scheme.

Chapter 6

Conclusions

In this work, we analysed the Bayesian framework and we applied it to deep neural networks. Such models have a very complex nature, especially in terms of number of parameters and their relations, hence it is not possible to apply the Bayes theorem in the canonical way and it is necessary to approximate the posterior distribution. To this end we selected variational inference because it produces an approximation that is neither too simplistic nor too slow.

Training Bayesian neural networks using the variational objective function, the ELBO, showed its own challenges. In fact it adds a number of new hyperparameters to the optimization problem, both local, like the shape of the prior and its parameters, and global, like the weight β of the Kullback–Leibler divergence. In particular, the prior distribution is a key choice in the variational inference process due to its regularization effect and the impact it has on the ELBO's value at convergence. For this reason, one of the main research directions of the field, is towards more flexible priors and structured variational inference (Carvalho et al., 2009; Ghosh and Doshi-Velez, 2017; Ghosh et al., 2018; Krueger et al., 2017; Lacoste et al., 2017).

Another major issue of the training is the variance of the gradient estimator of the loss function. A large variance leads to slow training and it may compromise the optimization process. In order to reduce this variance we used the pathwise gradient estimator and the local reparameterization trick which improved notably the speed of the training. The research community proposed different solutions for this problem, among these it is worth citing control variates which add a term to the gradients that does not change the expectation but decreases the variance (Johnson and Zhang, 2013; Paisley et al., 2012; Ranganath et al., 2014; Wang et al., 2013).

Bayesian neural networks with variational inference showed to be able to perform very well, almost reaching state-of-the-art performances of point estimates models. The added value of uncertainty modeling comes at the cost of an increase of both the training and the prediction time, hence it may not always be worth using such an approach against the standard one. However, variational inference itself is a trade-off both in terms of time and of quality of the posterior distribution. In fact, other alternatives exist to implement Bayesian neural networks, some are more accurate but require more computing power, i.e. MCMC methods, while others are faster but approximate the posterior in a different way, i.e. MC dropout.

Finally, the uncertainty measures derived from Bayesian neural networks have proved to be a real advantage for two different reasons: the first is the possibility to discard bad predictions of the model, therefore improving its general predictive power; the second is the capacity to analyze datasets to discover out-of-distribution samples.

The listed challenges and advances are the core of the new shift in deep learning research towards a new formulation of neural networks that still has to be improved but is already able to solve a new set of problems coming from continuously advancing real world applications.

References

Anderson, J. R. and Peterson, C. (1987). A mean field theory learning algorithm for neural networks. *Complex Systems*, 1:995–1019.

Ando, D. M., McLean, C., and Berndl, M. (2017). Improving phenotypic measurements in high-content imaging screens. *bioRxiv*, page 161422.

Azevedo-Filho, A. and Shachter, R. D. (1994). Laplace's method approximations for probabilistic inference in belief networks with continuous variables. In *Uncertainty Proceedings 1994*, pages 28–36. Elsevier.

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.

Beskos, A., Pillai, N., Roberts, G., Sanz-Serna, J.-M., Stuart, A., et al. (2013). Optimal tuning of the hybrid Monte Carlo algorithm. *Bernoulli*, 19(5A):1501–1534.

Betancourt, M. (2017). A conceptual introduction to Hamiltonian Monte Carlo. *arXiv preprint arXiv:1701.02434*.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg.

Blei, D. M., Kucukelbir, A., and McAuliffe, J. D. (2017). Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877.

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

Bos, C. S. (2002). A comparison of marginal likelihood computation methods. In *Compstat*, pages 111–116. Springer.

Bulatov, Y. (2011). notMNIST dataset. *http://yaroslavvb.blogspot.com/2011/09/notmnist-dataset.html*.

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.

Cao, C., Liu, F., Tan, H., Song, D., Shu, W., Li, W., Zhou, Y., Bo, X., and Xie, Z. (2018). Deep learning and its applications in biomedicine. *Genomics, proteomics & bioinformatics*, 16(1):17–32.

Carpenter, A. E., Jones, T. R., Lamprecht, M. R., Clarke, C., Kang, I. H., Friman, O., Guertin, D. A., Chang, J. H., Lindquist, R. A., Moffat, J., et al. (2006). Cellprofiler: image analysis software for identifying and quantifying cell phenotypes. *Genome biology*, 7(10):R100.

Caruana, R., Lawrence, S., and Giles, C. L. (2001). Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. In *Advances in neural information processing systems*, pages 402–408.

Carvalho, C. M., Polson, N. G., and Scott, J. G. (2009). Handling sparsity via the horseshoe. In *Artificial Intelligence and Statistics*, pages 73–80.

Chib, S. and Greenberg, E. (1995). Understanding the Metropolis-Hastings algorithm. *The american statistician*, 49(4):327–335.

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

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314.

Denker, J. S. and Lecun, Y. (1991). Transforming neural-net output levels to probability distributions. In *Advances in neural information processing systems*, pages 853–859.

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

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.

Gal, Y. (2016). Uncertainty in deep learning. *University of Cambridge*.

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

Ghosh, S. and Doshi-Velez, F. (2017). Model selection in bayesian neural networks via horseshoe priors. *arXiv preprint arXiv:1705.10388*.

Ghosh, S., Yao, J., and Doshi-Velez, F. (2018). Structured variational learning of bayesian neural networks with horseshoe priors. *arXiv preprint arXiv:1806.05975*.

Girolami, M. and Calderhead, B. (2011). Riemann manifold Langevin and Hamiltonian Monte Carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(2):123–214.

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.

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

Hastings, W. K. (1970). Monte Carlo sampling methods using Markov chains and their applications.

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

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

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

Jensen, J. L. W. V. (1906). Sur les fonctions convexes et les inégalités entre les valeurs moyennes. *Acta mathematica*, 30(1):175–193.

Johnson, R. and Zhang, T. (2013). Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in neural information processing systems*, pages 315–323.

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.

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

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

Kohavi, R. et al. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, volume 14, pages 1137–1145. Montreal, Canada.

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.

Krogh, A. and Hertz, J. A. (1992). A simple weight decay can improve generalization. In *Advances in neural information processing systems*, pages 950–957.

Krueger, D., Huang, C.-W., Islam, R., Turner, R., Lacoste, A., and Courville, A. (2017). Bayesian hypernetworks. *arXiv preprint arXiv:1710.04759*.

Kullback, S. and Leibler, R. A. (1951). On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86.

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*.

Lacoste, A., Boquet, T., Rostamzadeh, N., Oreshki, B., Chung, W., and Krueger, D. (2017). Deep prior. *arXiv preprint arXiv:1712.05016*.

Laplace, P. S. (1986). Memoir on the probability of the causes of events. *Statistical Science*, 1(3):364–378.

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

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

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

LeCun, Y., Jackel, L., Bottou, L., Brunot, A., Cortes, C., Denker, J., Drucker, H., Guyon, I., Muller, U., Sackinger, E., et al. (1995). Comparison of learning algorithms for handwritten digit recognition. In *International conference on artificial neural networks*, volume 60, pages 53–60. Perth, Australia.

Lever, J., Krzywinski, M., and Altman, N. (2016). Points of significance: model selection and overfitting.

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., et al. (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, V., Sokolnicki, K. L., and Carpenter, A. E. (2012). Annotated high-throughput microscopy image sets for validation. *Nature methods*, 9(7):637–637.

Logan, D. J. and Carpenter, A. E. (2010). Screening cellular feature measurements for image-based assay development. *Journal of biomolecular screening*, 15(7):840–846.

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.

MacKay, D. J. (1992). The evidence framework applied to classification networks. *Neural computation*, 4(5):720–736.

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, 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.

Miotto, R., Wang, F., Wang, S., Jiang, X., and Dudley, J. T. (2017). Deep learning for healthcare: review, opportunities and challenges. *Briefings in bioinformatics*, 19(6):1236–1246.

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, R. M. (1993). Probabilistic inference using Markov chain Monte Carlo methods.

Neal, R. M. (2012). *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media.

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

Nettleton, D. F., Orriols-Puig, A., and Fornells, A. (2010). A study of the effect of different types of noise on the precision of supervised learning techniques. *Artificial intelligence review*, 33(4):275–306.

Opper, M. and Winther, O. (1997). A mean field algorithm for bayes learning in large feed-forward neural networks. In *Advances in Neural Information Processing Systems*, pages 225–231.

Paisley, J., Blei, D., and Jordan, M. (2012). Variational bayesian inference with stochastic search. *arXiv preprint arXiv:1206.6430*.

Pawlowski, N., Caicedo, J. C., Singh, S., Carpenter, A. E., and Storkey, A. (2016). Automating morphological profiling with generic deep convolutional networks. *BioRxiv*, page 085118.

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

Rasmussen, C. E. (2004). Gaussian processes in machine learning. In *Advanced lectures on machine learning*, pages 63–71. Springer.

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

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386.

Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1985). Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, 323(6088):533.

Russell, S. J. and Norvig, P. (2016). *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural networks*, 61:85–117.

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

Simard, P. Y., Steinkraus, D., and Platt, J. C. (2003). Best practices for convolutional neural networks applied to visual document analysis. IEEE.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288.

Wainberg, M., Merico, D., Delong, A., and Frey, B. J. (2018). Deep learning in biomedicine. *Nature biotechnology*, 36(9):829.

Wainwright, M. J., Jordan, M. I., et al. (2008). Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1–2):1–305.

Wang, C., Chen, X., Smola, A. J., and Xing, E. P. (2013). Variance reduction for stochastic gradient optimization. In *Advances in Neural Information Processing Systems*, pages 181–189.

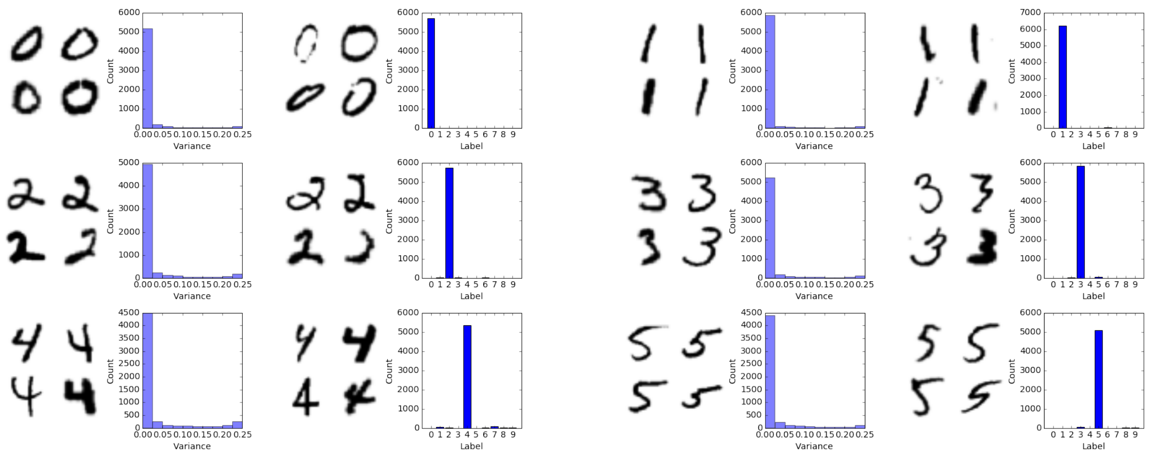
Wong, T.-T. (2015). Performance evaluation of classification algorithms by k-fold and leave-one-out cross validation. *Pattern Recognition*, 48(9):2839–2846.

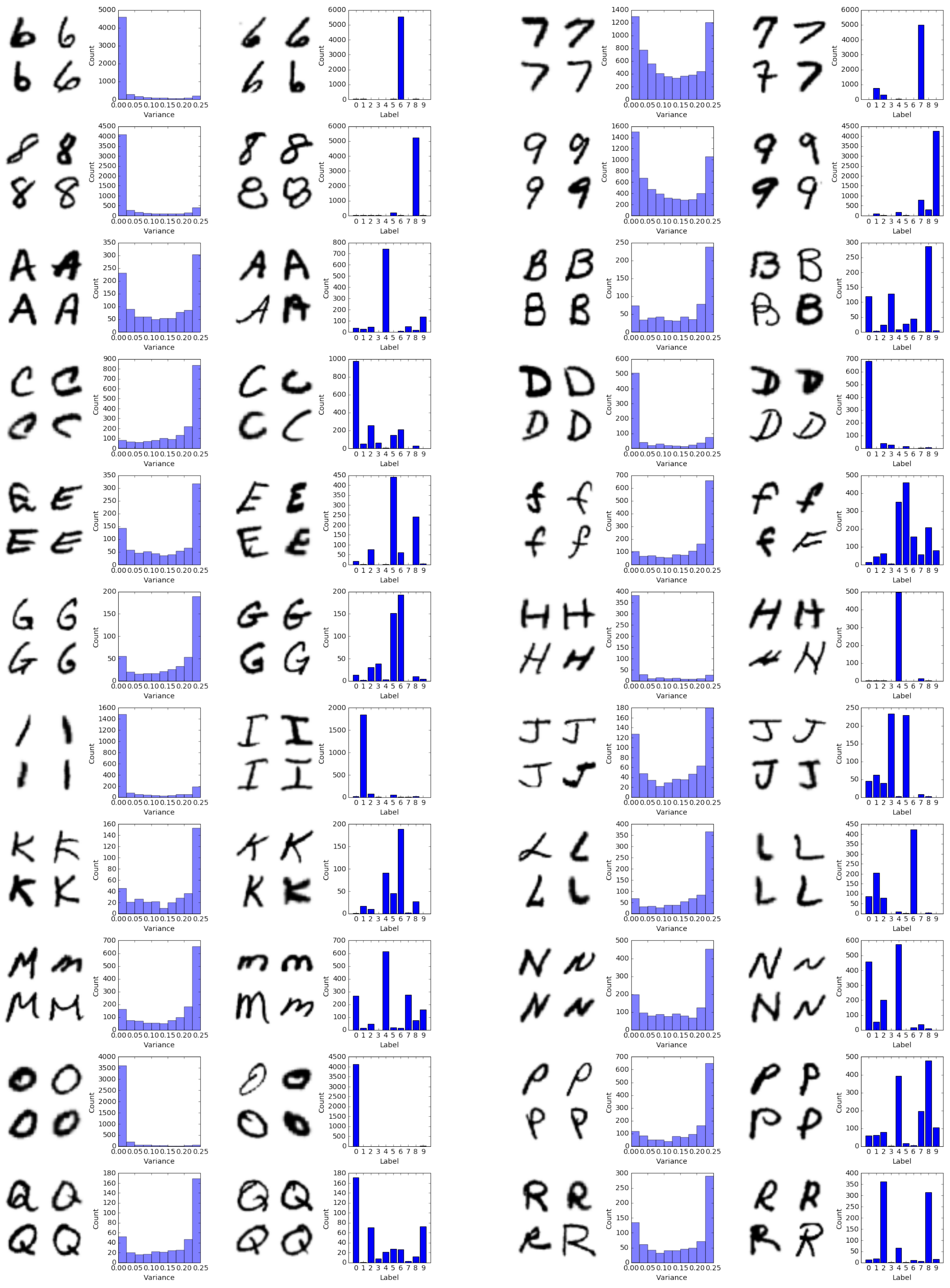
EMNIST predictive variance

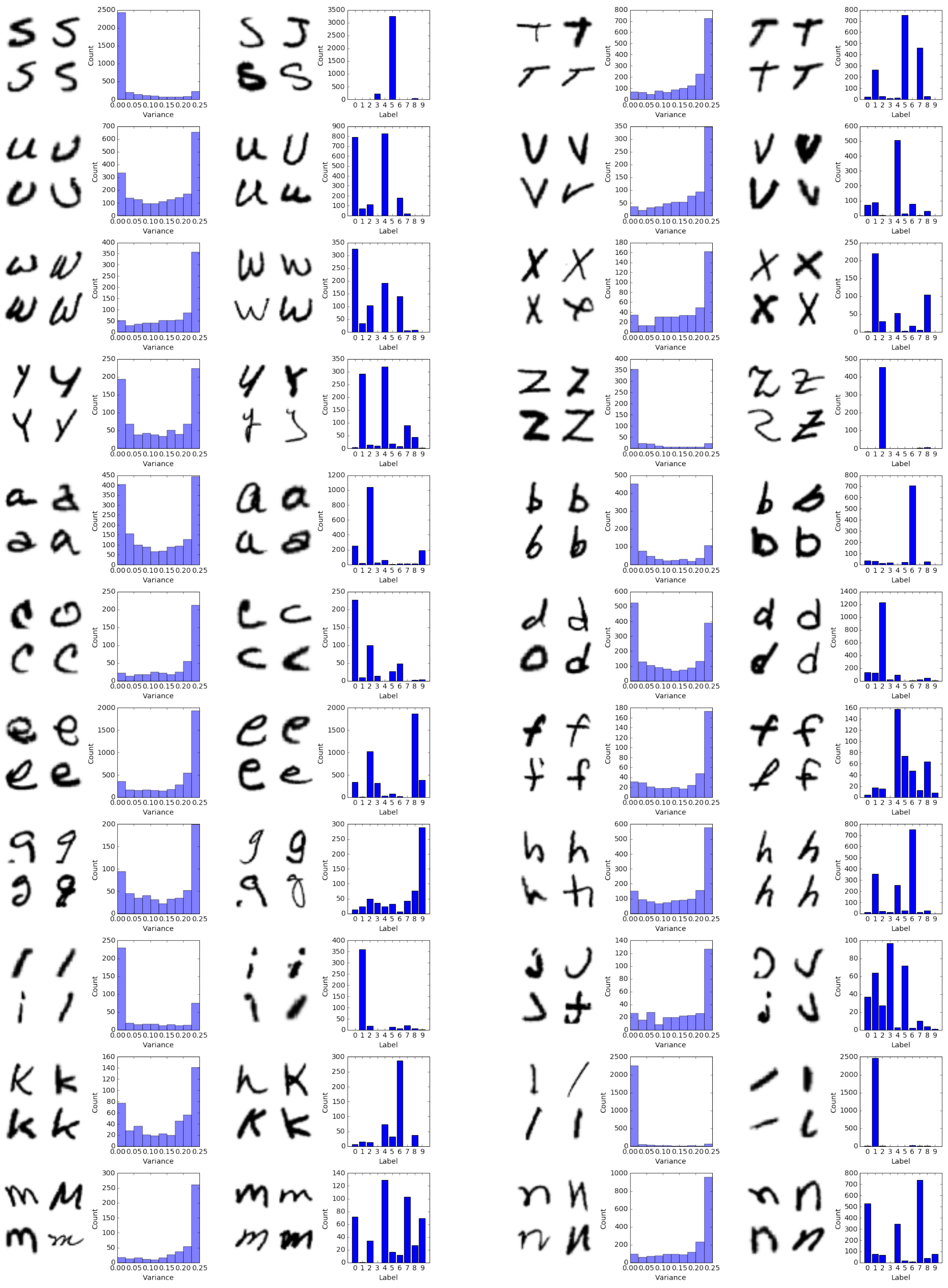
In this appendix we illustrate the predictive variance for each class of the EMNIST classes. This helps to understand which classes are familiar to the model because they have mainly low variance, and which classes it recognizes as new.

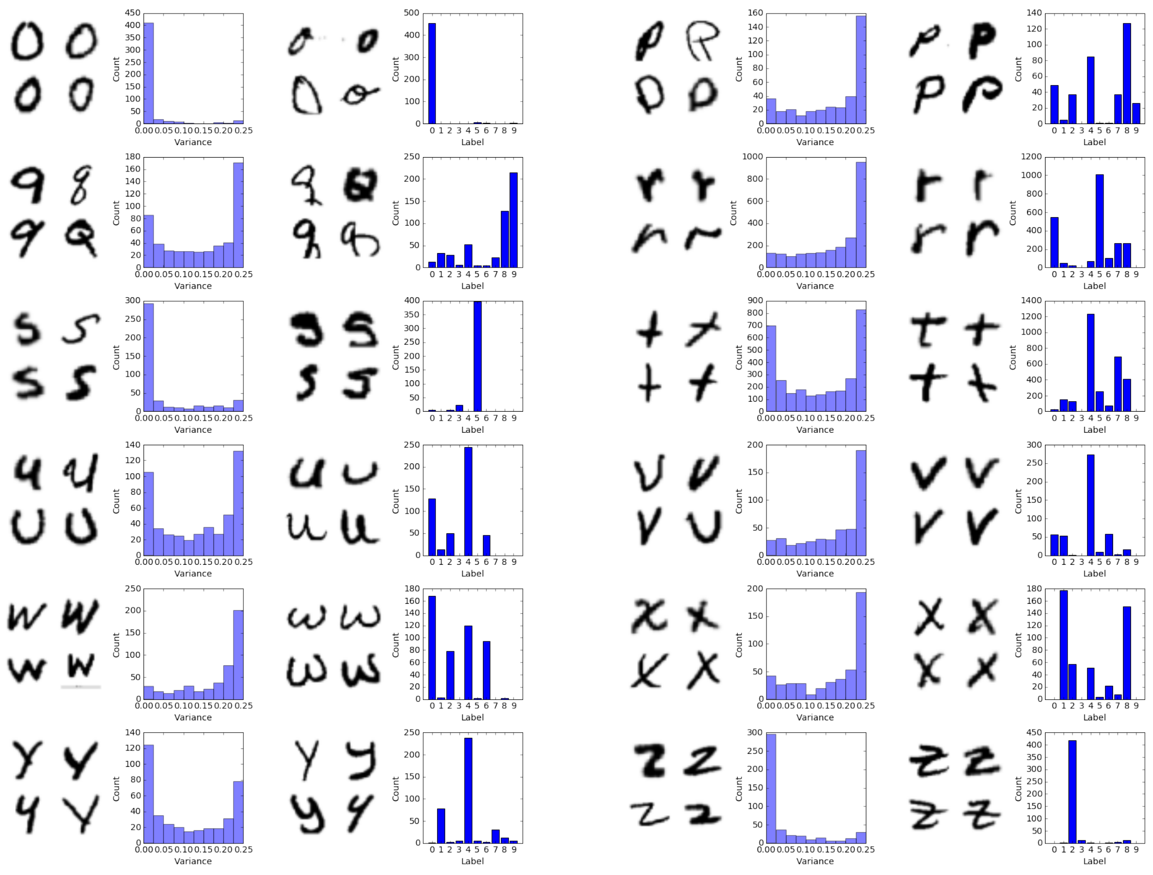
Next to each variance histogram we show some images which have been predicted with low and high uncertainty to see if the model is there is discriminating the images inside each class. For example, well written digits from ambiguous ones, but also letters that look like numbers from letters that don't.

Finally, we added the count plots with the corresponding predictions for each class. This helps to validate any hypothesis regarding the predictive variance. For example, number images are predicted mostly with low variance, and the corresponding count plot has a great count corresponding to the right class. In the same way, letters that look like numbers have very high counts on the corresponding class while the other letters have more heterogeneous counts.



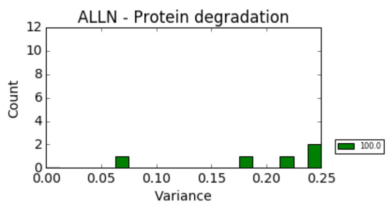

Count
Variance
Label

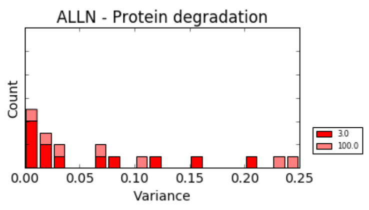

Count
Variance
Label
0.00 0.05 0.10 0.15 0.20 0.25
0 1 2 3 4 5 6 7 8 9

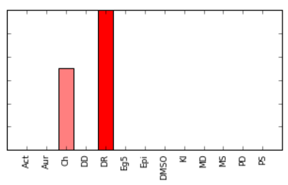

Count
Variance
Label
0.00 0.05 0.10 0.15 0.20 0.25
0 1 2 3 4 5 6 7 8 9

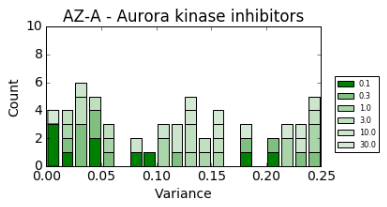
BBBC021 Leave-One-Compound-Out validation

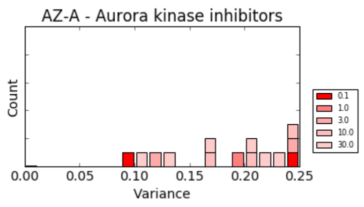
In this appendix we illustrate the predictive variance for each compound of the BBBC021 dataset when performing Leave-One-Compound-Out cross validation. We show the histograms of both the correct and wrong predictions and, when the latter is not empty, we also display the count plot of the predictions corresponding to each concentration that is colored in the same way as in the variance histogram.

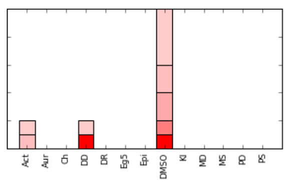


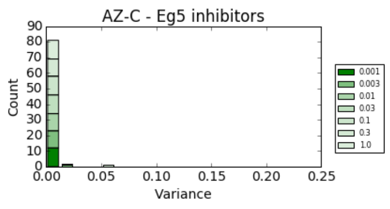


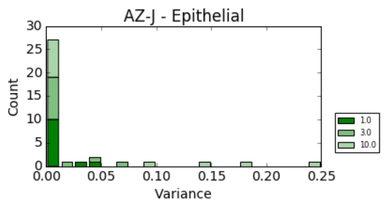


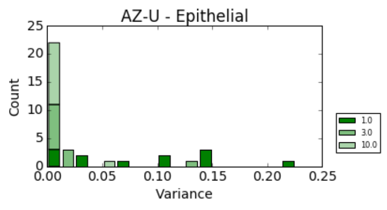


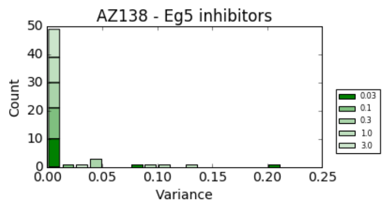


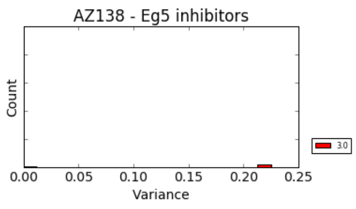


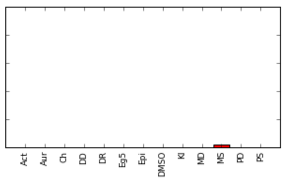


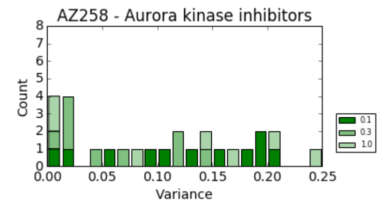


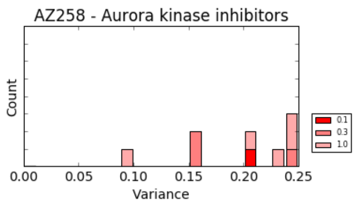

AZ-U - Epithelial
Count
Variance
1.0
3.0
10.0

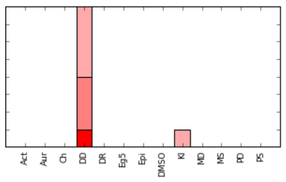

AZ138 - Eg5 inhibitors
Count
Variance
0.03
0.1
0.3
1.0
3.0

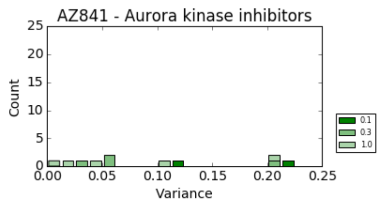

AZ138 - Eg5 inhibitors
Count
Variance
3.0

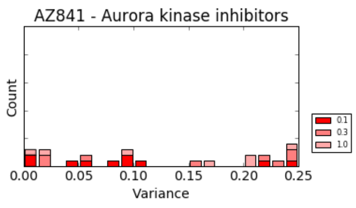

Act
Aur
Ch
DD
DR
Eg5
Epi
DMSO
KI
MD
MS
PD
PS

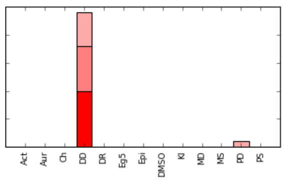

AZ258 - Aurora kinase inhibitors
Count
Variance
0.1
0.3
1.0

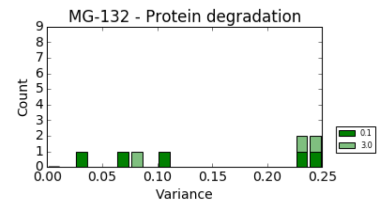

AZ258 - Aurora kinase inhibitors
Count
Variance
0.1
0.3
1.0

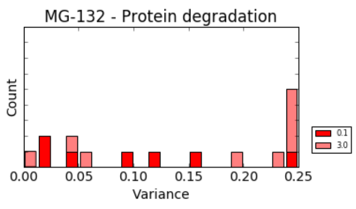

Act
Aur
Ch
DD
DR
Eg5
Epi
DMSO
KI
MD
MS
PD
PS

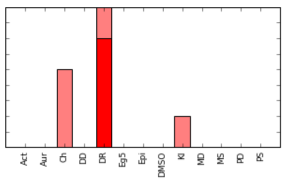

AZ841 - Aurora kinase inhibitors
Count
Variance
0.1
0.3
1.0

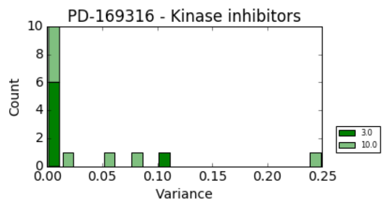

AZ841 - Aurora kinase inhibitors
Count
Variance
0.1
0.3
1.0

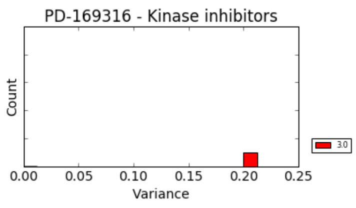

Act
Aur
Ch
DD
DR
Eg5
Epi
DMSO
KI
MD
MS
PD
PS

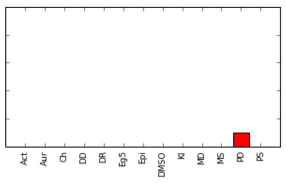

MG-132 - Protein degradation
Count
Variance
0.1
3.0

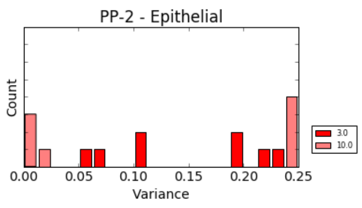

MG-132 - Protein degradation
Count
Variance
0.1
3.0

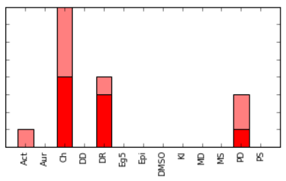

Act
Aur
Ch
DD
DR
Eg5
Epi
DMSO
KI
MD
MS
PD
PS

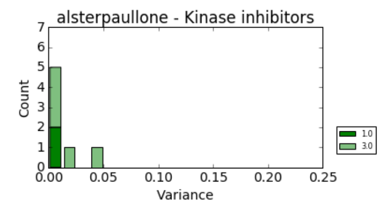

PD-169316 - Kinase inhibitors
Count
Variance
3.0
10.0

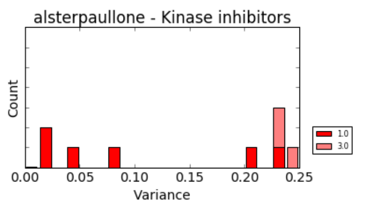

PD-169316 - Kinase inhibitors
Count
Variance
3.0

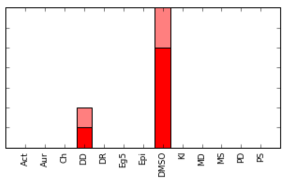

Act
Aur
Ch
DD
DR
Eg5
Epi
DMSO
KI
MD
MS
PD
PS

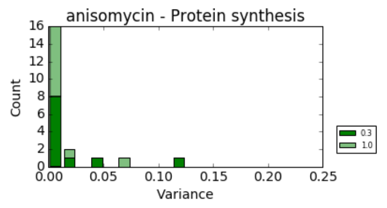

PP-2 - Epithelial
Count
Variance
3.0
10.0

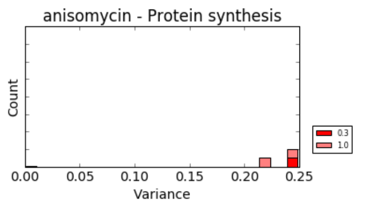

Act
Aur
Ch
DD
DR
Eg5
Epi
DMSO
KI
MD
MS
PD
PS

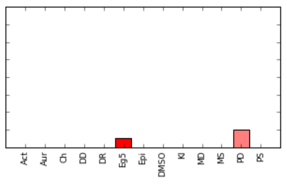

alsterpaullone - Kinase inhibitors
Count
Variance
1.0
3.0

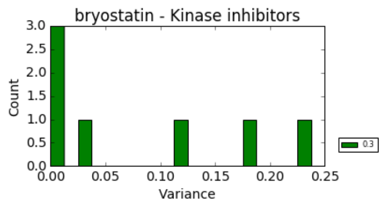

alsterpaullone - Kinase inhibitors
Count
Variance
1.0
3.0

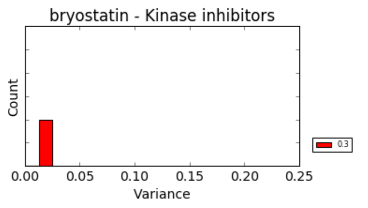

Act
Aur
Ch
DD
DR
Eg5
Epi
DMSO
KI
MD
MS
PD
PS

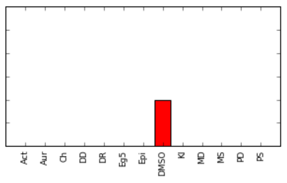

anisomycin - Protein synthesis
Count
Variance
0.3
1.0

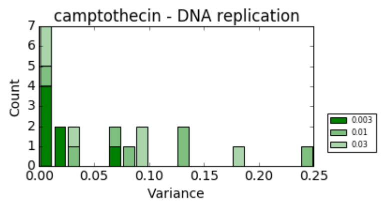

anisomycin - Protein synthesis
Count
Variance
0.3
1.0

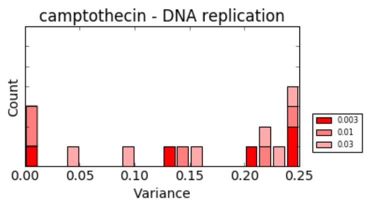

Act
Aur
Ch
DD
DR
Eg5
Epi
DMSO
KI
MD
MS
PD
PS

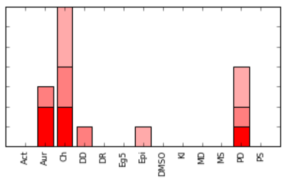

bryostatin - Kinase inhibitors
Count
Variance
0.3

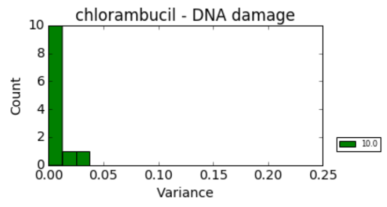

bryostatin - Kinase inhibitors
Count
Variance
0.3

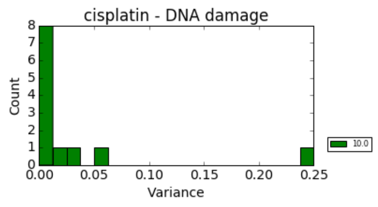

Act
Aur
Ch
DD
DR
Eg5
Epi
DMSO
KI
MD
MS
PD
PS

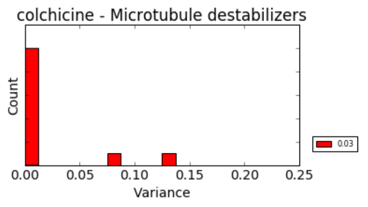

camptothecin - DNA replication
Count
Variance
0.003
0.01
0.03

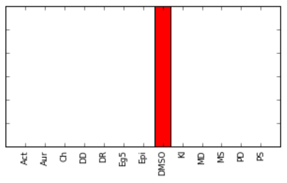

camptothecin - DNA replication
Count
Variance
0.003
0.01
0.03

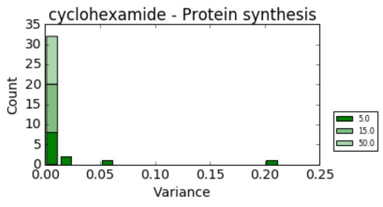

Act
Aur
Ch
DD
DR
Eg5
Epi
DMSO
KI
MD
MS
PD
PS

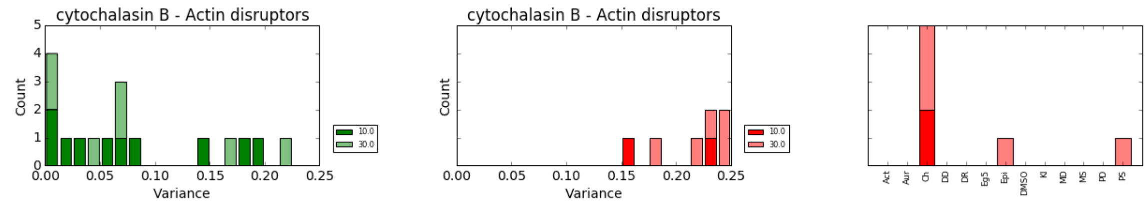

chlorambucil - DNA damage
Count
Variance
10.0

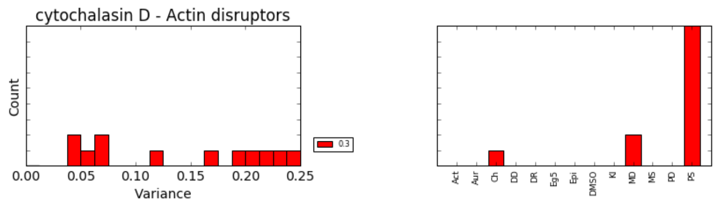

cisplatin - DNA damage
Count
Variance
10.0

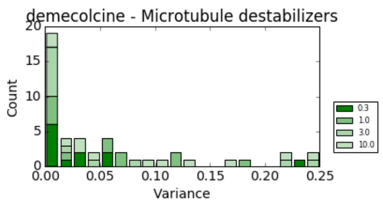

colchicine - Microtubule destabilizers
Count
Variance
0.03

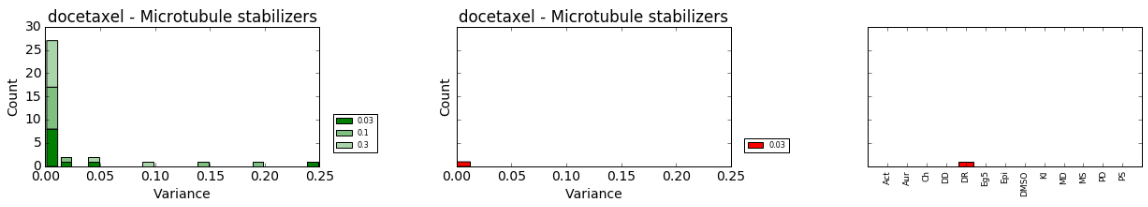

Act
Aur
Ch
DD
DR
Eg5
Epi
DMSO
KI
MD
MS
PD
PS

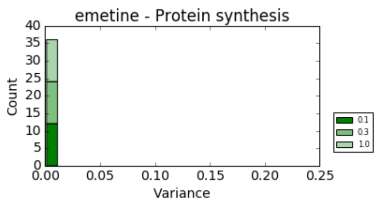

cyclohexamide - Protein synthesis
Count
Variance
5.0
15.0
50.0

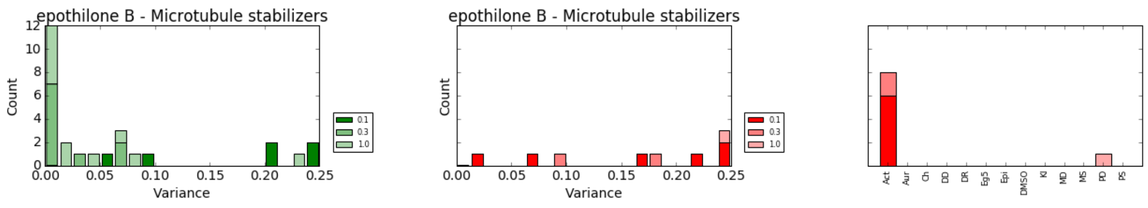

cytochalasin B - Actin disruptors
Count
Variance
10.0
30.0
cytochalasin B - Actin disruptors
Count
Variance
10.0
30.0
Act
Aur
Ch
DD
DR
Eg5
Epi
DMSO
KI
MD
MS
PD
PS

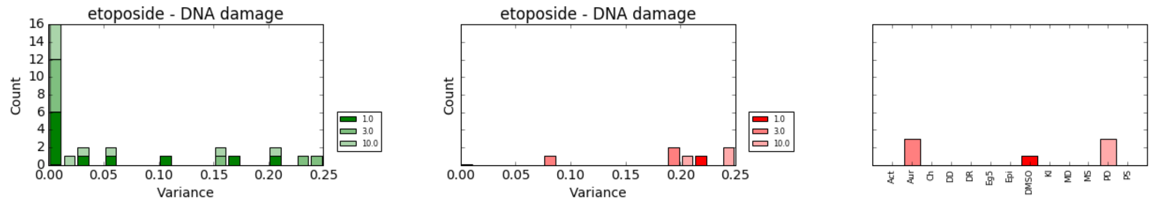

cytochalasin D - Actin disruptors
Count
Variance
0.3
Act
Aur
Ch
DD
DR
Eg5
Epi
DMSO
KI
MD
MS
PD
PS

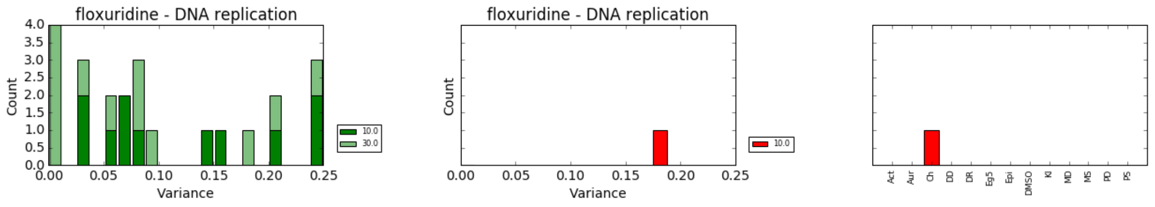

demecolcine - Microtubule destabilizers
Count
Variance
0.3
1.0
3.0
10.0

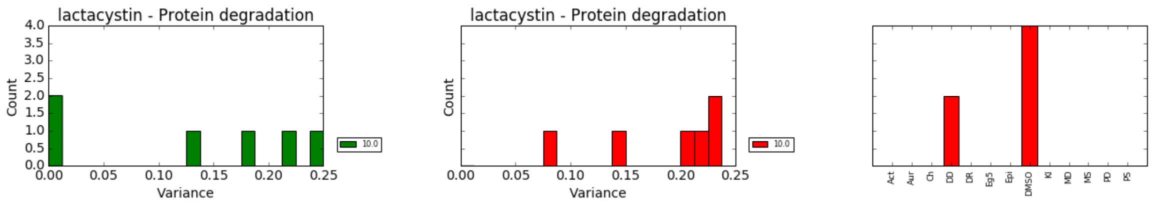

docetaxel - Microtubule stabilizers
Count
Variance
0.03
0.1
0.3
docetaxel - Microtubule stabilizers
Count
Variance
0.03
Act
Aur
Ch
DD
DR
Eg5
Epi
DMSO
KI
MD
MS
PD
PS

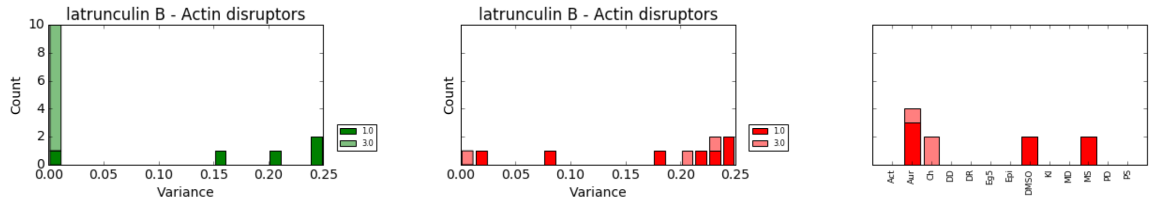

emetine - Protein synthesis
Count
Variance
0.1
0.3
1.0

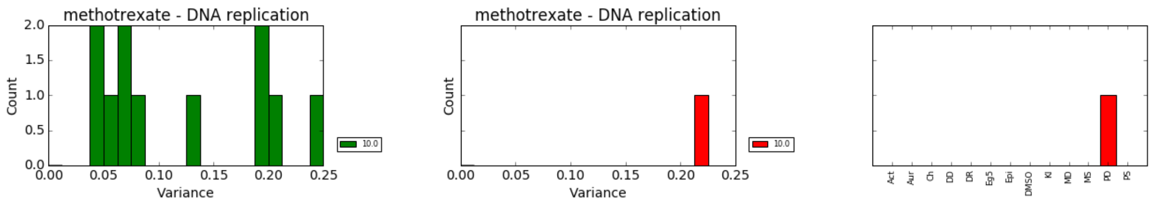

epothilone B - Microtubule stabilizers
Count
Variance
0.1
0.3
1.0
epothilone B - Microtubule stabilizers
Count
Variance
0.1
0.3
1.0
Act
Aur
Ch
DD
DR
Eg5
Epi
DMSO
KI
MD
MS
PD
PS

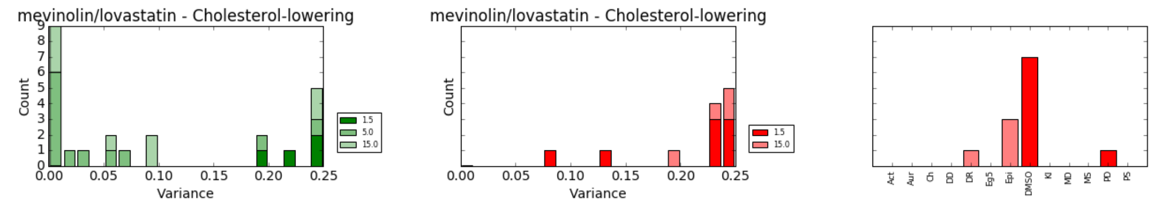

etoposide - DNA damage
Count
Variance
1.0
3.0
10.0
etoposide - DNA damage
Count
Variance
1.0
3.0
10.0
Act
Aur
Ch
DD
DR
Eg5
Epi
DMSO
Kl
MD
MS
PD
PS

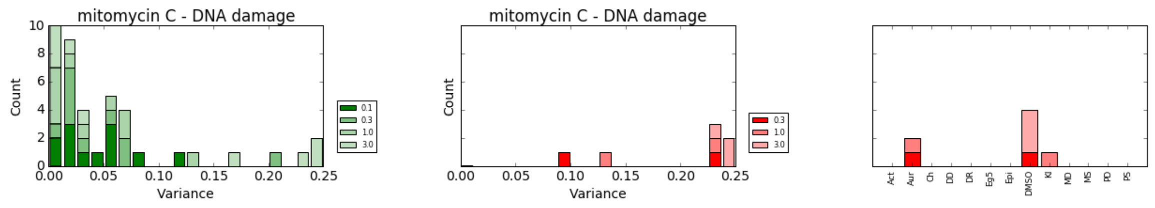

floxuridine - DNA replication
Count
Variance
10.0
30.0
floxuridine - DNA replication
Count
Variance
10.0
Act
Aur
Ch
DD
DR
Eg5
Epi
DMSO
Kl
MD
MS
PD
PS

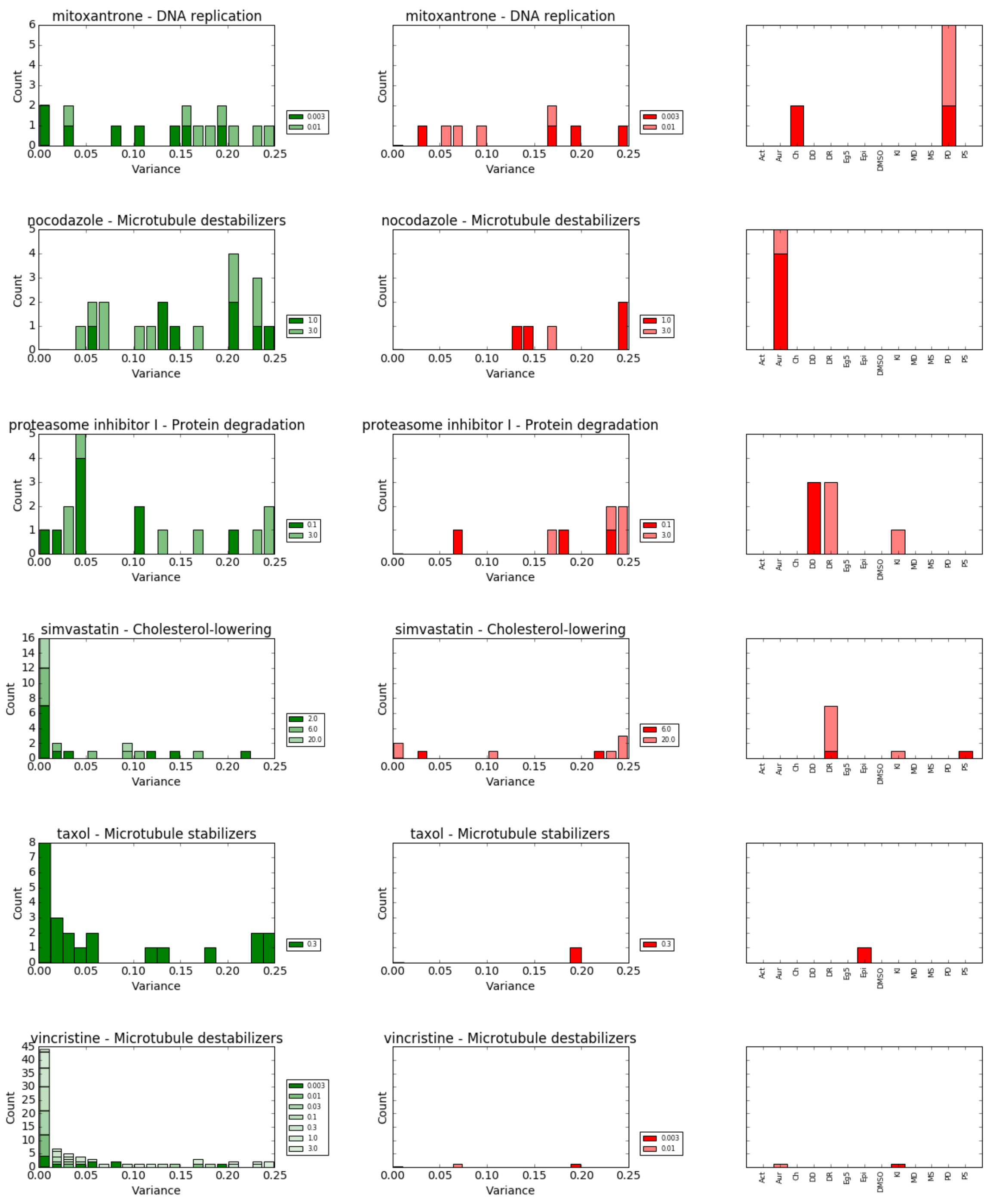

lactacystin - Protein degradation
Count
Variance
10.0
lactacystin - Protein degradation
Count
Variance
10.0
Act
Aur
Ch
DD
DR
Eg5
Epi
DMSO
Kl
MD
MS
PD
PS


latrunculin B - Actin disruptors
Count
Variance
1.0
3.0
latrunculin B - Actin disruptors
Count
Variance
1.0
3.0
Act
Aur
Ch
DD
DR
Eg5
Epi
DMSO
Kl
MD
MS
PD
PS


methotrexate - DNA replication
Count
Variance
10.0
methotrexate - DNA replication
Count
Variance
10.0
Act
Aur
Ch
DD
DR
Eg5
Epi
DMSO
Kl
MD
MS
PD
PS

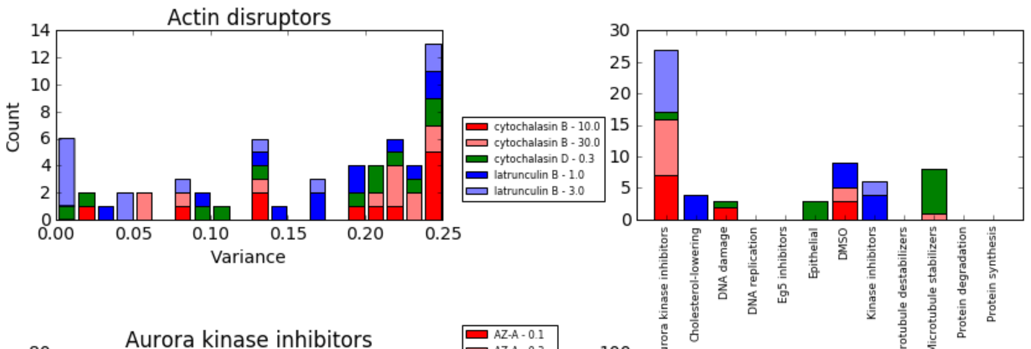

mevinolin/lovastatin - Cholesterol-lowering
Count
Variance
1.5
5.0
15.0
mevinolin/lovastatin - Cholesterol-lowering
Count
Variance
1.5
15.0
Act
Aur
Ch
DD
DR
Eg5
Epi
DMSO
Kl
MD
MS
PD
PS

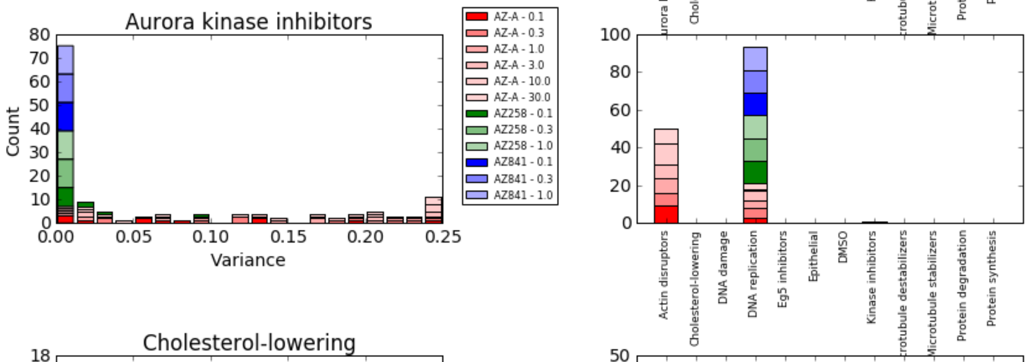

mitomycin C - DNA damage
Count
Variance
0.1
0.3
1.0
3.0
mitomycin C - DNA damage
Count
Variance
0.3
1.0
3.0
Act
Aur
Ch
DD
DR
Eg5
Epi
DMSO
Kl
MD
MS
PD
PS

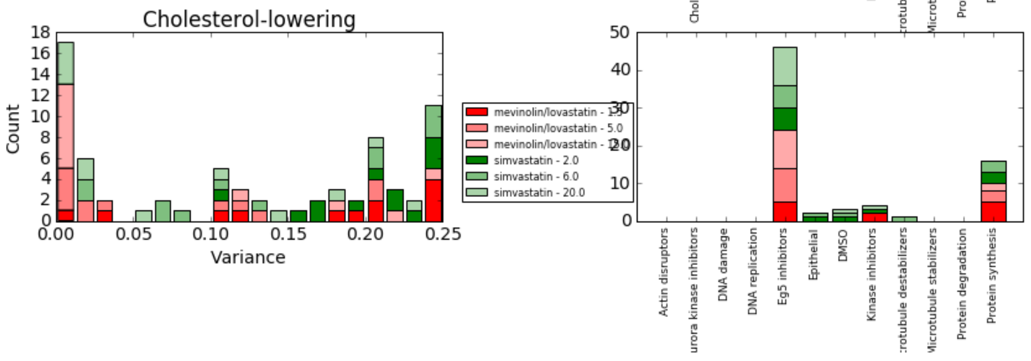

mitoxantrone - DNA replication
nocodazole - Microtubule destabilizers
proteasome inhibitor I - Protein degradation
simvastatin - Cholesterol-lowering
taxol - Microtubule stabilizers
vincristine - Microtubule destabilizers
Count
Variance
0.003
0.01
0.03
0.1
0.3
1.0
2.0
3.0
6.0
20.0
Act
Aur
Ch
DD
DR
Eg5
Epi
DMSO
KI
MD
MS
PD
PS

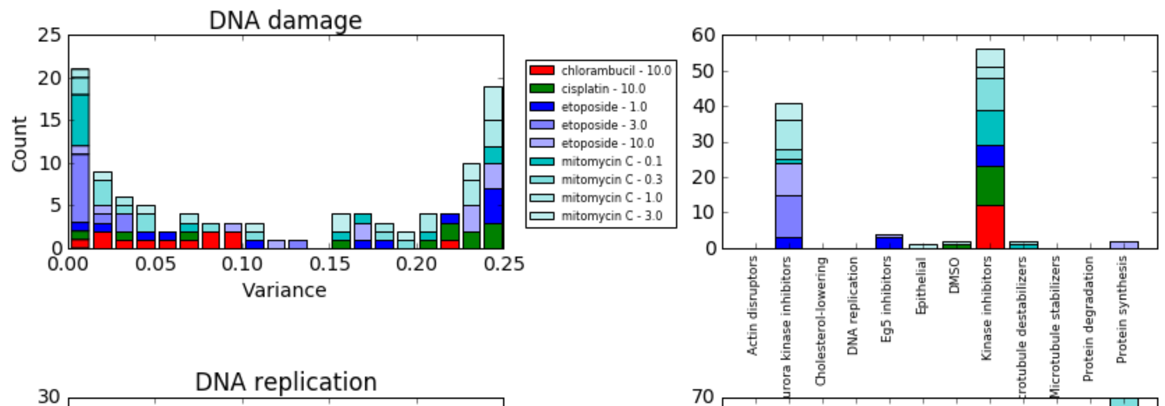
BBBC021 Leave-One-MOA-Out validation

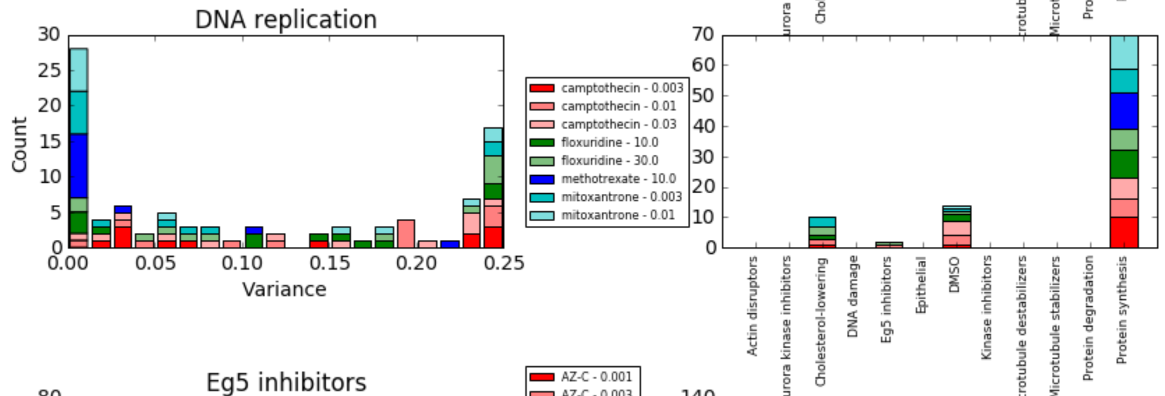
In this appendix we illustrate the predictive variance for each MOA of the BBBC021 dataset when performing Leave-One-MOA-Out cross validation. Next to the histogram of the predictive variance we illustrate the count plot of the predictions corresponding to each compound that is colored in the same way in the variance histogram.

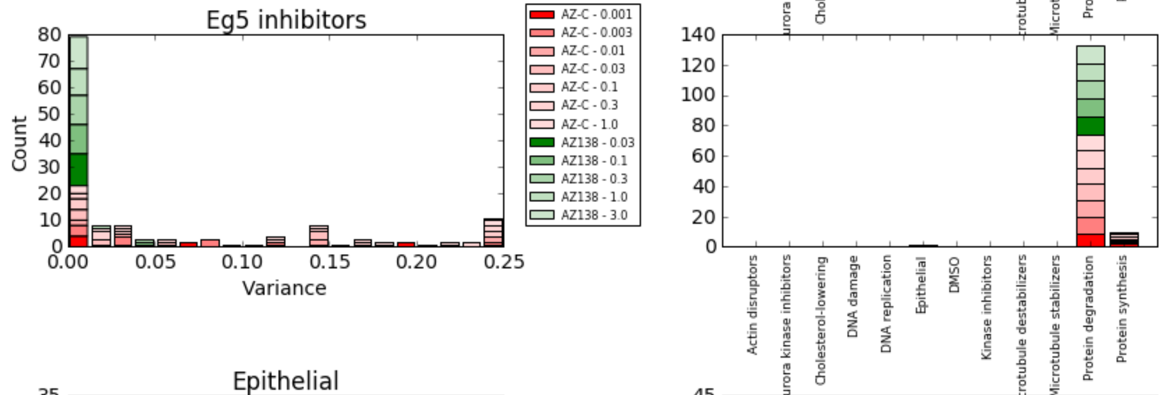


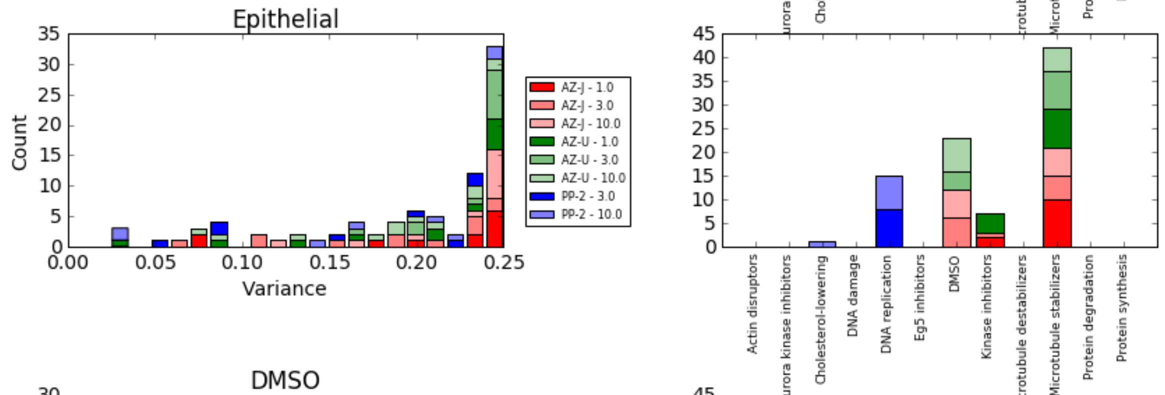


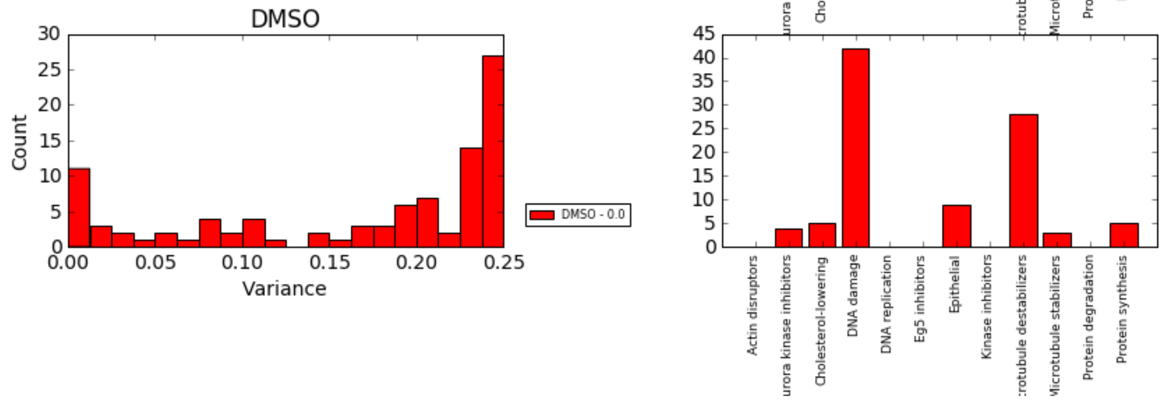


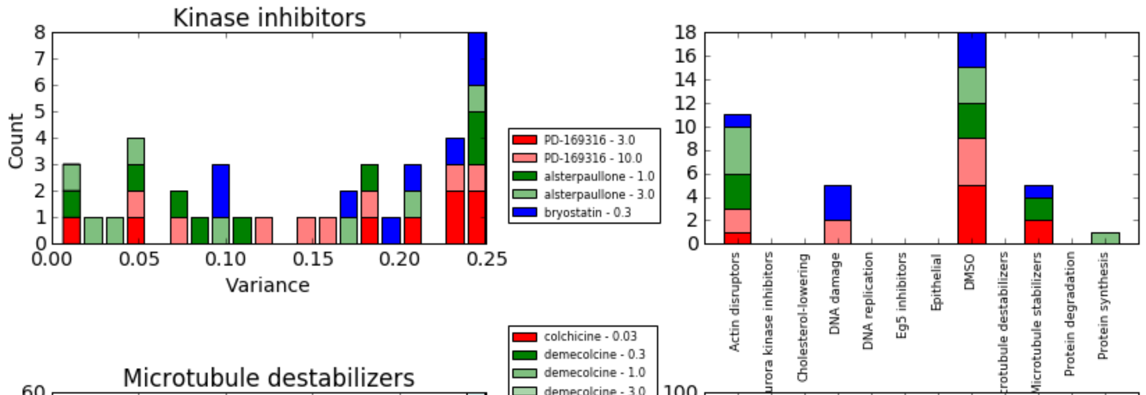

DNA damage
Count
Variance
chlorambucil - 10.0
cisplatin - 10.0
etoposide - 1.0
etoposide - 3.0
etoposide - 10.0
mitomycin C - 0.1
mitomycin C - 0.3
mitomycin C - 1.0
mitomycin C - 3.0
Actin disruptors
urora kinase inhibitors
Cholesterol-lowering
DNA replication
Eg5 inhibitors
Epithelial
DMSO
Kinase inhibitors
rotubule destabilizers
Microtubule stabilizers
Protein degradation
Protein synthesis

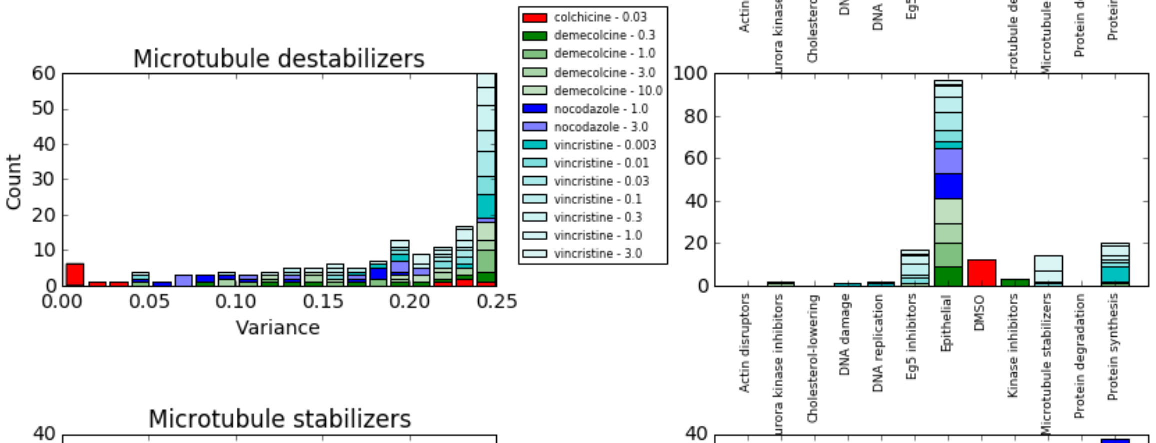

DNA replication
Count
Variance
camptothecin - 0.003
camptothecin - 0.01
camptothecin - 0.03
floxuridine - 10.0
floxuridine - 30.0
methotrexate - 10.0
mitoxantrone - 0.003
mitoxantrone - 0.01
Actin disruptors
urora kinase inhibitors
Cholesterol-lowering
DNA damage
Eg5 inhibitors
Epithelial
DMSO
Kinase inhibitors
rotubule destabilizers
Microtubule stabilizers
Protein degradation
Protein synthesis

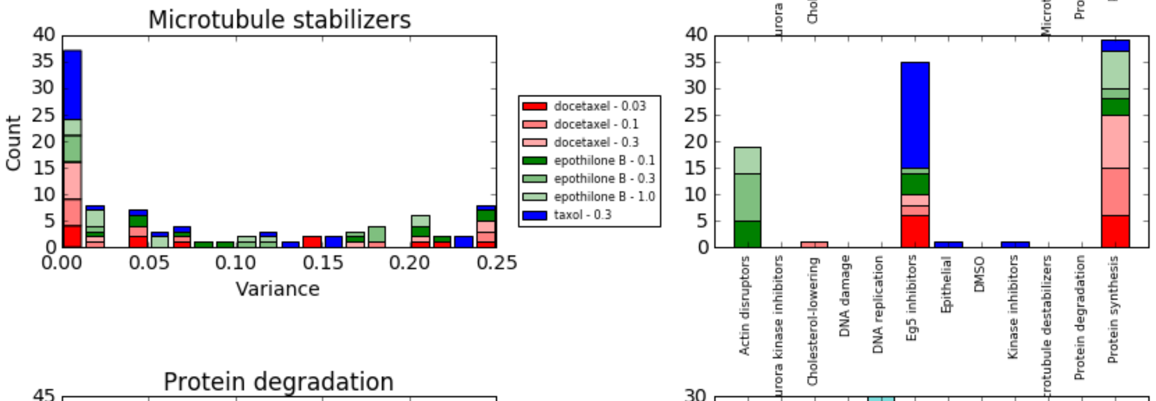

Eg5 inhibitors
Count
Variance
AZ-C - 0.001
AZ-C - 0.003
AZ-C - 0.01
AZ-C - 0.03
AZ-C - 0.1
AZ-C - 0.3
AZ-C - 1.0
AZ138 - 0.03
AZ138 - 0.1
AZ138 - 0.3
AZ138 - 1.0
AZ138 - 3.0
Actin disruptors
urora kinase inhibitors
Cholesterol-lowering
DNA damage
DNA replication
Epithelial
DMSO
Kinase inhibitors
rotubule destabilizers
Microtubule stabilizers
Protein degradation
Protein synthesis

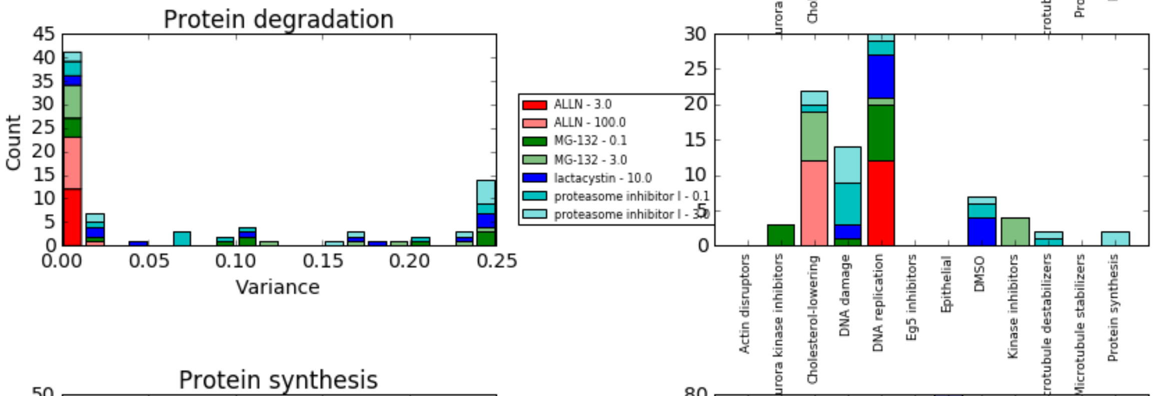

Epithelial
Count
Variance
AZ-J - 1.0
AZ-J - 3.0
AZ-J - 10.0
AZ-U - 1.0
AZ-U - 3.0
AZ-U - 10.0
PP-2 - 3.0
PP-2 - 10.0
Actin disruptors
urora kinase inhibitors
Cholesterol-lowering
DNA damage
DNA replication
Eg5 inhibitors
DMSO
Kinase inhibitors
rotubule destabilizers
Microtubule stabilizers
Protein degradation
Protein synthesis

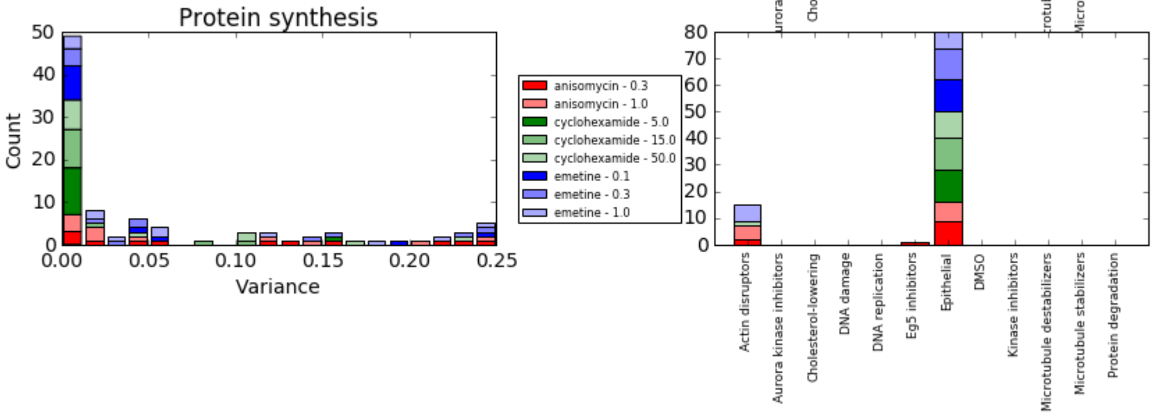

DMSO
Count
Variance
DMSO - 0.0
Actin disruptors
urora kinase inhibitors
Cholesterol-lowering
DNA damage
DNA replication
Eg5 inhibitors
Epithelial
Kinase inhibitors
rotubule destabilizers
Microtubule stabilizers
Protein degradation
Protein synthesis


Kinase inhibitors
Count
Variance
PD-169316 - 3.0
PD-169316 - 10.0
alsterpaullone - 1.0
alsterpaullone - 3.0
bryostatin - 0.3
Actin disruptors
urora kinase inhibitors
Cholesterol-lowering
DNA damage
DNA replication
Eg5 inhibitors
Epithelial
DMSO
rotubule destabilizers
Microtubule stabilizers
Protein degradation
Protein synthesis


Microtubule destabilizers
Count
Variance
colchicine - 0.03
demecolcine - 0.3
demecolcine - 1.0
demecolcine - 3.0
demecolcine - 10.0
nocodazole - 1.0
nocodazole - 3.0
vincristine - 0.003
vincristine - 0.01
vincristine - 0.03
vincristine - 0.1
vincristine - 0.3
vincristine - 1.0
vincristine - 3.0
Actin disruptors
urora kinase inhibitors
Cholesterol-lowering
DNA damage
DNA replication
Eg5 inhibitors
Epithelial
DMSO
Kinase inhibitors
Microtubule stabilizers
Protein degradation
Protein synthesis


Microtubule stabilizers
Count
Variance
docetaxel - 0.03
docetaxel - 0.1
docetaxel - 0.3
epothilone B - 0.1
epothilone B - 0.3
epothilone B - 1.0
taxol - 0.3
Actin disruptors
urora kinase inhibitors
Cholesterol-lowering
DNA damage
DNA replication
Eg5 inhibitors
Epithelial
DMSO
Kinase inhibitors
rotubule destabilizers
Protein degradation
Protein synthesis


Protein degradation
Count
Variance
ALLN - 3.0
ALLN - 100.0
MG-132 - 0.1
MG-132 - 3.0
lactacystin - 10.0
proteasome inhibitor I - 0.1
proteasome inhibitor I - 3.0
Actin disruptors
urora kinase inhibitors
Cholesterol-lowering
DNA damage
DNA replication
Eg5 inhibitors
Epithelial
DMSO
Kinase inhibitors
rotubule destabilizers
Microtubule stabilizers
Protein synthesis


Protein synthesis
Count
Variance
anisomycin - 0.3
anisomycin - 1.0
cyclohexamide - 5.0
cyclohexamide - 15.0
cyclohexamide - 50.0
emetine - 0.1
emetine - 0.3
emetine - 1.0
Actin disruptors
Aurora kinase inhibitors
Cholesterol-lowering
DNA damage
DNA replication
Eg5 inhibitors
Epithelial
DMSO
Kinase inhibitors
Microtubule destabilizers
Microtubule stabilizers
Protein degradation