Bayesian neural networks to analyse hyperspectral datasets using uncertainty metrics

Adrián Alcolea*, Javier Resano*

***Abstract*—Machine learning techniques, and specifically neural networks, have proved to be very useful tools for image classification tasks. Nevertheless, measuring the reliability of these networks and calibrating them accurately is very complex. This is even more complex in a field like hyperspectral imaging, where labelled data are scarce and difficult to generate. Bayesian neural networks (BNNs) allow to obtain uncertainty metrics related to the data processed (aleatoric), and to the uncertainty generated by the model selected (epistemic). On this work we will demonstrate the utility of BNNs by analysing the uncertainty metrics obtained by a BNN over five of the most used hyperspectral images datasets. In addition we will illustrate how these metrics can be used for several practical applications such as identifying predictions that do not reach the required level of accuracy, detecting mislabelling in the dataset, or identifying when the predictions are affected by the increase of the level of noise in the input data.**

***Index Terms*—Bayesian networks, uncertainty, aleatoric uncertainty, epistemic uncertainty, hyperspectral images.**

I. Introduction

MACHINE Learning techniques have made spectacular advances in recent years. The improvements and refinement of the models used, together with hardware platforms that allow the exploration of increasingly complex models, trained on larger data sets, have enabled a vast improvement in the accuracy of the models, and opened up new fields of application.

In remote sensing, neural networks (NNs) and convolutional neural networks (CNNs) [1] have demonstrated to be one of the most useful tools for hyperspectral image classification, with many recent relevant publications [2], [3], [4], [5]. They provide state-of-the-art results, but they have some limitations. A NN model will provide very good results if it is trained with data similar to the data that it will process when deployed. However, if the training set includes data with high noise level, or it includes mislabelled data, or it is unbalanced, or the model, once in operation, receives inputs with different features, or different formats from those used in training, the predictions will not be reliable. This problem is not easy to solve since Machine Learning is based on the learn-from-data paradigm. Hence, the quality of the results of a NN model relies on the quality of the training data. However, we can deal with this problem in a more efficient way by enriching the models based on NNs or CNNs so that, together with their predictions, they report uncertainty metrics that can be used to identify these issues.

*Group of Computer Architecture (GaZ), Engineering Research Institute of Aragon (I3A), University of Zaragoza, e-mails: alcolea@unizar.es, jresano@unizar.es

In this manuscript, we will evaluate these possibilities for the problem of pixel classification of hyperspectral images. For this purpose, we will use a simple model based on NNs and upgrade it into a bayesian neural network (BNN), i.e. a model that combines neural network with Bayesian inference. BNNs use distributions to model weights and outputs. Hence, they will not generate a constant output for a given input, but a distribution, that can be analysed to measure the prediction uncertainty. Moreover, it allows to differentiate between the uncertainty generated by the model itself, and the uncertainty generated by the input data. This approach can be applied to NNs and CNNs in a similar way. We will use a simple NN model, instead of a large CNN because our goal is not to achieve the maximum accuracy, but to explore the utility of using BNNs instead of conventional NNs. This is orthogonal to the size of the model, hence, following the recommendations of the Green AI paradigm [6], we selected a model that can be trained fast with low energy consumption (we trained hundreds of different models during this research), and provides results that can be easily replicated by any researcher, without the need for specific hardware platforms to train the model. We will demonstrate the utility of our approach, with several experiments applied to some of the most frequently used hyperspectral data sets. First, we will propose a scheme to test if the model is properly calibrated. Second, we will show how the uncertainty metric can be used to achieve a requested level of accuracy by discarding inputs that have very high levels of uncertainty. The goal is not to artificially increase the accuracy, but to allow our model to answer ”I don't know”. In fact, there are many situations in which saying, ”I don't know,” is the correct answer. For example, the network may receive a pixel that do not belong to any of the categories, or a pixel which include a mix of them. It is critical to be able to identify such cases, either to simply ignore those outputs, mark them as unreliable outputs, use an alternative method to classify them, or to try to understand why our model fails in these situations. These situations occur in the data sets analysed, because neither the labelling, nor our models, nor the training process are perfect, but they will be more frequent if the models are used in the real world, because all kinds of new spectral signatures will appear in the input that will not belong to any of the trained categories, and therefore should not be classified.. Third, we will analyse the uncertainty of the different categories of each data set. Finally, we will test the model response to two different experiments: training a new network mixing the labels of two classes, and the introduction of white noise during inference. The code used for the experiments and the trained models are available in a public repository so that they can serve as a baseline for applying this technique to other problems [7].

A. *Motivational example*

Using a simple neuron as an example we can easily illustrate the impact of bayesian models in our prediction abilities. For that, we are going to train a neuron using the training data represented on Table I that includes two different datasets, each of which has three training data for the same input $(0.5, 0.5)$. In both cases, for that input, a trained neuron will generate a function with an output about 0.7. A possible solution, is presented in Equation 1, where both weights are 0.7 and bias is 0, since it is the output that minimise the error for both data sets. But the inputs of datasets 1 and 2 are very different. In the first case, the training data is consistent, and the outputs are very similar for the input $(0.5, 0.5)$, while in the second case the outputs have a large variance. If the models generate the same output in both cases we will lose that information.

TABLE I
BAYESIAN NEURON EXAMPLE INPUTS.

	Dataset 1			Dataset 2		
Data	Input1	Input2	Output	Input1	Input2	Output
First	0.5	0.5	0.6	0.5	0.5	0.2
Second	0.5	0.5	0.7	0.5	0.5	0.7
Third	0.5	0.5	0.8	0.5	0.5	1.2

$$O(I_1, I_2) = ReLU(I_1 \times 0.7 + I_2 \times 0.7 + 0) \qquad (1)$$

The weights and bias of a bayesian neuron are not constant values, but distributions, as shown in Figure 1, so every time we call the function we will receive a different weight value according with that distribution. If we train this bayesian neuron for datasets 1 and 2, the mean value of the distributions on both models will be similar, and with the same value that in the previous cases, 0.7, but the deviation will be higher for case 2, because the distribution is wider, therefore, if we perform several inference passes over the two models, the output average will be around 0.7 on both of them, but we will be able to observe an important difference on the deviation of the outputs. For the first data set, all the inference passes will provide similar results, while for the second case, the output distribution will be wider, and the results will have larger divergence. Uncertainty metrics make it possible to differentiate between these two situations.

II. Related work

As explained in the introduction, NNs have demonstrated to be one of the most accurate techniques for hyperspectral image classification. CNNs can exploit both the spectral and the spatial information of hyperspectral image applying convolution filters and pooling operations to the input data. Based on the output of the previous layer, each new layer generate more complex feature vectors, which are finally processed by one or several layers of fully connected neurons. Deep CNNs achieve state of the art classification accuracy in general for image classification, and in particular for hyperspectral images. In [2] the authors analyse different machine learning methods evaluating its accuracy, its size, and the computations required. In this analysis CNNs achieve the best accuracy. The works described in [3], [8], [4], [5] are examples of state-of-the-art results using CNNs.

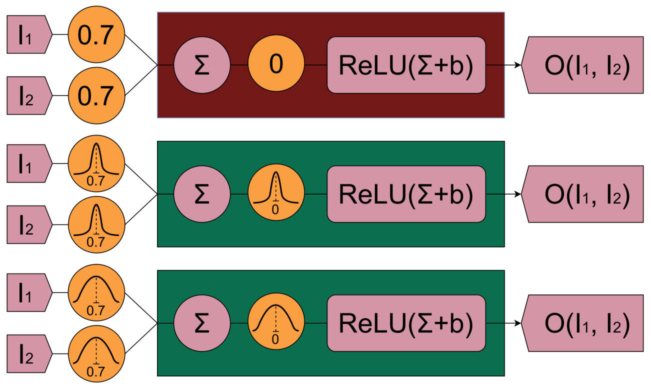

Fig. 1. weights of normal neuron for both datasets (top) versus bayesian neuron on dataset 1 (centre) and dataset 2 (bottom).

However, using deep CNNs for hyperspectral is very complex, as it requires a great amount of labelled training data for tuning their large number of parameters, and in remote sensing labelled samples are very difficult and expensive to collect. Moreover, the high dimensionality of the hyperspectral data further complicates the setting of the deep model parameters. This may lead to overfitting in many networks. These networks will provide good results for the training data, and for the test data if it is very similar, but their results are not generalisable for new data. This is a very complex issue, the training data can be improved by adding more samples, or by using data augmentation techniques to generate additional training samples by performing several transformation to the original data set [9].

In this context, BNNs can provide added value with their ability to generate uncertainty estimations. However, very few works have explored the possibilities of using BNNs for hyperspectral pixel classification. The most representative is the work presented in [10]. Their objective is to achieve higher classification accuracies for situations with scarce labelled data. To this end, they propose to combine BNNs with active learning for the problem of hyperspectral pixel classification. They start the training process with very few labelled pixels and then use a BNN model to select and introduce in the training set some of the unlabelled pixels in order to improve the results. Their Bayesian approach is dropout based, following the main idea described in [11].

Another work that uses BNNs for hyperspectral images was presented in [12], although in this case, they are not used to classify pixels, but to monitor water quality from an unmanned aerial vehicle. In this case, they use a BNN because it allows using a single model to generate multiple outputs, thus emulating an ensemble of models working together, but they do not measure uncertainty.

These previous works focus on increasing accuracy, which is, of course, an important goal. However, if we only look at accuracy, the advantages of using BNNs are not clear, since any result obtained with a BNN could also be obtained with an ensemble of NNs that carries out the same computations. Nevertheless, in order to use NNs for real-world problems, the reliability of the predictions is as important as accuracy itself [13], and the uncertainty metrics offered by BNNs can be the way to improve reliability [14], [15]. Therefore, we believe that it is very important to analyse the possibilities offered by BNNs to achieve more reliable neural networks in this field. Sometimes, for classification problems, softmax layers results, i.e. the output of the last layer of the NN, have been wrongly interpreted as a measure of the uncertainty of the output, but as explained in [16], [17], they cannot be interpreted in such way because their probabilities distribution are not relative to the uncertainty of the model on their predictions. Therefore, specific uncertainty metrics are needed. A good example of the utility of these metrics is [17]. In this work, the authors use the uncertainty metrics of a BNN to prove that multi-spectral images offer better reliability on classification problems than RGB images.

In this paper, we want to analyse the opportunities offered by the uncertainty metrics provided by BNNs for the problem of hyperspectral image pixel classification. To illustrate them, we developed some experiments that demonstrate their added value.

III. Bayesian Networks and Uncertainty Quantification

Neural networks (NNs) have proved to be very powerful techniques to perform image classification tasks, and they also achieved great results with hyperspectral images as we exposed in last section. Nevertheless, there are still some generalisation issues, such as overfitting, that could specially affect those areas where the datasets are scarce and difficult to generate, as remote sensing hyperspectral imaging. Probabilistic models such as bayesian neural networks allow us to analyse the model uncertainty for a given prediction due to their stochastic behaviour.

Bayesian approaches model probability distributions to express the uncertainty over the unobserved data. For that, the model starts with a prior distribution of probabilities that updates according to the observed data. After training, the generated model should represent the uncertainty about each parameter value. So, instead of weights, the calculated values to represent the network are random variables initialised with a prior distribution $p(\mathbf{w})$, and the training will consist on calculating the posterior $p(\mathbf{w}|\mathbf{D})$, where $\mathbf{D}$ represents the observed data $\mathbf{D} = \{\mathbf{y}, \mathbf{x}\}$ [18], [19], [20].

However, computing the exact posterior $p(\mathbf{w}|\mathbf{D})$ for big and complicated models as NNs is intractable. Hence, bayesian neural networks are based in approximate models. During the training phase, variational Bayesian methods are used to approximate intractable integrals. One of the most commonly used is variational inference (VI), which tries to approximate the parameters ϕ of a variational distribution $q_\phi(\mathbf{w})$ to minimise its Kullback-Leibler (KL) with $p(\mathbf{w}|\mathbf{D})$ [18], [20]. Theoretically, distributions could have any shape, but it is impossible to analyse the search space for that case. To solve this issue, only symmetric and tractable distributions are used. Gaussian functions are typically used for this purpose because they are defined with only two parameters, which simplifies the training process, and allows BNNs to be more compact. Therefore, compared to regular NNs, BNNs based on Gaussian functions will have twice as many parameters. This is an important overhead for large models, but, as explained in the previous section, a BNN can be seen as an ensemble of models, since each inference will generate a different output. Large NN ensembles are frequently used to reduce the variance and improve the accuracy of the output. Hence, a single BNN can replace the complete ensemble and lead to important reductions in the size of the model.

In this paper we want to analyse the uncertainty of the predictions. Since BNNs are probabilistic models, executing T stochastic inferences over the same test dataset, will provide T different output predictions. This allows to measure the model uncertainty, but also to separately quantify aleatoric uncertainty, which is related to the quality of the dataset itself, and epistemic uncertainty, which is related to our trained model. For that analysis we will use the predictive entropy ($\mathbb{H}$), expected entropy ($\mathbb{E}_p$) and mutual information (MI) of the T resultant predictions.

Let K be the number of classes of the dataset and $\{\mathbf{c}_k\}_{k=1}^{K}$ the set of class labels. Predictive entropy ($\mathbb{H}$) represents the overall uncertainty of the model within the range $[0, \log(K)]$. We already defined $\mathbf{D}$ as the observed data, $\mathbf{w}$ as the calculated variables of the model and T as the number of stochastic forward passes. The value of $\mathbb{H}$ is given by Eq. 2, where $a_t = p(\mathbf{y} = \mathbf{c}_k|\mathbf{x}, \mathbf{w}_t)$ [13].

$$\mathbb{H}(\mathbf{y}|\mathbf{x}, \mathbf{D}) := -\sum_{k=1}^{K}\left(\frac{1}{T}\sum_{t=1}^{T} a_t\right)\log\left(\frac{1}{T}\sum_{t=1}^{T} a_t\right) \quad (2)$$

Expected entropy ($\mathbb{E}_p$), given by Eq. 3, represents the aleatoric uncertainty. This variable can be used to analyse the dataset characteristics and will be a baseline to determine how training data alterations can affect the model [13]. High aleatoric uncertainty values indicate ambiguities in the data that can be caused by noise, mislabelled or misrepresented data, overlapping categories, or any other issues that make the dataset difficult to learn.

$$\mathbb{E}_{p(\mathbf{w}|\mathbf{D})}\left[\mathbb{H}(\mathbf{y}|\mathbf{x}, \mathbf{w})\right] := \frac{1}{T}\sum_{t=1}^{T}\left(-\sum_{k=1}^{K} a_t \log(a_t)\right) \quad (3)$$

Mutual information (MI) captures the epistemic uncertainty of the model and will give us information about uncertainty generated by model. The MI value is given by Eq. 4 and we will often refer to it as $\mathbb{H} - \mathbb{E}_p$ [13]. High epistemic uncertainty values indicate that the model is not properly trained or that it is not the right model.

$$MI(\mathbf{y}, \mathbf{w}|\mathbf{x}, \mathbf{D}) := \mathbb{H}(\mathbf{y}|\mathbf{x}, \mathbf{D}) - \mathbb{E}_{p(\mathbf{w}|\mathbf{D})}\left[\mathbb{H}(\mathbf{y}|\mathbf{x}, \mathbf{w})\right] \quad (4)$$

IV. Datasets and model characteristics

A. Datasets

We selected for our tests five of the most used hyperspectral datasets, Botswana (BO) [21], Indian Pines (IP) [21], Kennedy Space Center (KSC) [21], Pavia University (PU) [21] and Salinas Valley (SV) [21]. Table II shows the number of classes, labelled pixels of each class and spectral bands of every dataset. For a visual idea of the classes distribution on the images, the ground truth of all of them is represented in Section V, in Figures 9 to 13.

TABLE II
Images characteristics.

	BO	IP	KSC	PU	SV
Number of classes	14	16	13	9	16
Class 0 px.	270	46	761	6631	2009
Class 1 px.	101	1428	243	18649	3726
Class 2 px.	251	830	256	2099	1976
Class 3 px.	215	237	252	3064	1394
Class 4 px.	269	483	161	1345	2678
Class 5 px.	269	730	229	5029	3959
Class 6 px.	259	28	105	1330	3579
Class 7 px.	203	478	431	3682	11271
Class 8 px.	314	20	520	947	6203
Class 9 px.	248	972	404	-	3278
Class 10 px.	305	2455	419	-	1068
Class 11 px.	181	593	503	-	1927
Class 12 px.	268	205	927	-	916
Class 13 px.	95	1265	-	-	1070
Class 14 px.	-	386	-	-	7268
Class 15 px.	-	93	-	-	1807
Total labelled px.	3248	10249	5211	42776	56975
Spectral bands	145	200	176	103	204

B. Model Training

For the purposes of this research we trained for every image a NN model: a Multilayer Perceptron (MLP) with two hidden layers, the first one with 32 nodes and the second one with 16 nodes. The framework used to generate and train the networks is TensorFlow [22] with DenseFlipout layers, which implement bayesian variational inference with Flipout estimator, and we used as *kd_function* parameter the *kl_divergence* function divided by the number of labelled pixels on the dataset and *ReLU* as the activation function [23], [24]. In a previous work [2], we performed a grid search to identify the optimal size of a neural network, among other models, for each different hyperspectral image, and we found that the best results, without overfitting, ranged from 80 neurons to 140 depending on the image. However, since the goal of this research is not to identify the optimal NN configuration, but to explore the additional possibilities offered by a Bayesian network, we selected a smaller model, with only 48 neurons because it achieved results almost as good as a model with twice as many neurons, but it can be trained much faster. We used the 50% of the pixels of each class for training, with 10% reserved for validation to detect overfitting, and the other 50% for testing. We used an initial learning rate of 0.01 for every dataset. We trained the models for 50000 epochs storing the intermediate states every 100 epochs. At the end, we analysed the accuracy obtained in each of those intermediate states with the validation set, and chosen the ones with the best accuracy results. Among them, we chose those with the lowest average uncertainty. And among those with similar uncertainties, we chose the model that had been trained the least number of epochs. The training information is summarised in Table III. We used *categorical_crossentropy* loss function on an Adam optimiser. During inference, we execute 100 bayesian passes to extract the uncertainty information and to average the predictions. The entire code is available on [7].

TABLE III
Training data.

				Accuracy		**Uncertainty**	
Image	Train %	LR	Epochs	Train	Test	Train	Test
BO	50%	0.01	17000	0.94	0.91	0.24	0.26
IP	50%	0.01	22000	0.90	0.86	0.33	0.35
KSC	50%	0.01	41000	0.93	0.92	0.24	0.26
PU	50%	0.01	1800	0.95	0.95	0.16	0.17
SV	50%	0.01	4000	0.93	0.93	0.17	0.17

C. Model Calibration

Before using a model, it is necessary to evaluate its quality. Typically, this is done by analysing the accuracy of the test results. However, accuracy should not be the only concern. If we want a reliable model, we must also verify that it is well calibrated, this can be evaluated using a Reliability Diagram [14], [25], [26]. To construct a diagram for the test results of a BNN we propose a simple scheme based on defining a bin for each range of output values (from 0 to 1). For example we can define 10 bins, to store values ranging from $[0\%, 10\%)$, from $[10\%, 20\%)$, and so on. The output values will be interpreted as probabilities, and the diagram provides feedback about the accuracy of theses values. For example, a value of 0.25 would indicate that there is a 25% chance that the class is correct. What our diagram tries to check is whether, on average, these probabilities correspond to what is observed when verifying the data obtained with the test data set. What we expect for a well calibrated model is that there will be a correspondence between the average accuracy of the elements assigned to a bin, and the range of values assigned to that bin. For example, the outputs assigned to the bin ranging from $[0\%, 10\%)$ should correspond to the correct class between 0% and 10% of the time.

On every stochastic pass, the network output for each pixel will be a distribution of K probabilities, one for each class. We then calculate the average value of the T stochastic passes, so we will have one final prediction of K probabilities for each pixel. These probabilities are grouped in bins. After that, each of them is evaluated, and we compute the accuracy of each bin as the number of correct predictions divided by the number of total predictions assigned to the bin.

Figure 2 shows the results of the calibration of our model for the five datasets. We can observe that the curve of every

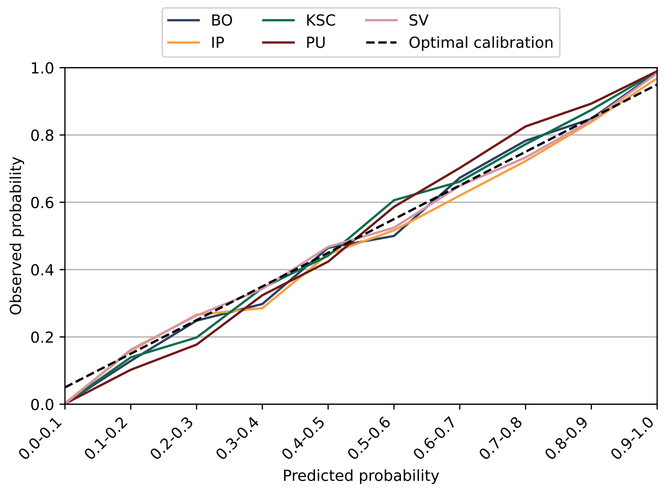

Fig. 2. Reliability Diagram.

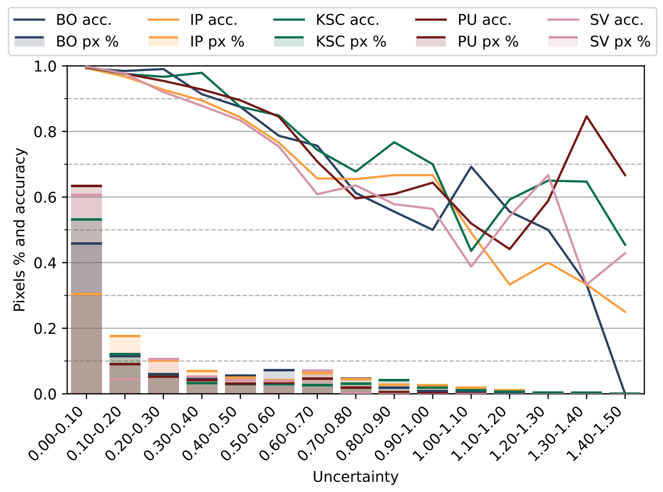

Fig. 3. Percentage of pixels and accuracy along uncertainty groups.

image is very close to a perfect calibration. From these results we can infer that the result of averaging the T stochastic passes generates a well calibrated model.

V. Experimental Results

A. Accuracy vs uncertainty

BNNs can be used to generate a stronger, and properly calibrated model, using the average of several stochastic passes, in the same way that ensembles of NNs are used for the same purpose. This is important, but it does not use one of the most interesting features of BNNs, the uncertainty information. In this experiment, we want to explore the relationship between uncertainty and accuracy. If there is correlation between them, once our network is already trained and tested, it will be possible to define an uncertainty threshold in order to filter out predictions with very high uncertainty values. As explained in the introduction, the objective is not to artificially increase the accuracy, but to allow our model to answer "I don't know". If the level of uncertainty exceeds the set threshold, it means that our network is not able to provide an output with the required quality. This can be due to many reasons. A clear example would be that the input does not belong to any of the trained categories, or is a mixture of several of them. Other more complex cases will be discussed later.

The upper part of Figure 3 depicts the relationship between accuracy and uncertainty for the five datasets. As we can observe in the chart, those outputs with less than 0.1 of uncertainty achieve almost 100% of accuracy, and those with uncertainty values between 0.3 and 0.4 still maintain an accuracy of more than 90% on every image. Therefore, we can adjust the uncertainty threshold to obtain the precision we need.

The distribution of pixels with respect to the uncertainty values is also very important because setting an uncertainty threshold that optimises the accuracy will not be useful if we discard most of the pixels. At the bottom of Figure 3 can be seen that most of the pixels have an uncertainty value of less than 0.1 and the distribution is clearly decreasing. It can also be seen that the distribution of pixels is quite different in each dataset. In the following experiment we want to analyse this in detail, examining each of its classes separately.

B. Class uncertainty

Figures 4 to 8 represent the uncertainty values ($\mathbb{H}$) on a comfortable scale divided into two categories: $\mathbb{E}_p$, the aleatoric uncertainty that measures the uncertainty for the input data, and $\mathbb{H} - \mathbb{E}_p$, the epistemic uncertainty that measures the uncertainty of the model. The average ($\mathbb{H}$) values are between 0.2 and 0.4, which is between 7.5% and 14.5% of their theoretical maximum, as explained in Section III, the uncertainty range is $[0, \log(K)]$, where K is the number of classes. We can also observe that in all the cases most of the uncertainty is due to the data. This information can be used to determine whether the selected model should be improved, either by changing it, or by training it for a longer period, or whether the results are constrained by the quality of the dataset itself. In this case using a deeper model, or training the model longer may improve the results, but only marginally, as the main source of uncertainty is the data used.

We can also observe that, on every image, there are significant differences between classes, meaning that some of them are more difficult to identify. That could be caused because there are classes with similar features, mislabelled pixels, or because the labelled pixels of a particular class are not enough to determine all their characteristics. For example, the uncertainty of class 8 of IP is 3.5 times higher than average, indicating that there is a very clear problem with the data in that class. In KSC several classes duplicate the average entropy. In the description of the dataset they already warned of this problem since they have identified that certain vegetation types have similar spectral signatures. In SV classes 7 and 14 are clearly harder to identify for the BNN than the rest of the classes, and the same applies to PU classes 2 and 4. This analysis also identifies which classes work particularly well, reporting small values in the uncertainty metrics. An

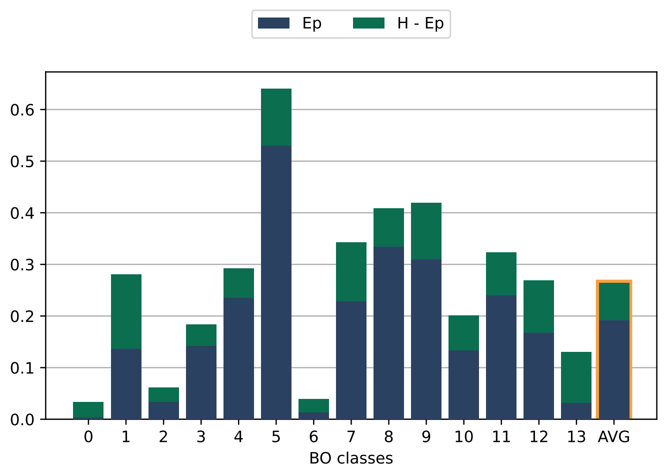

Fig. 4. BO class uncertainty.

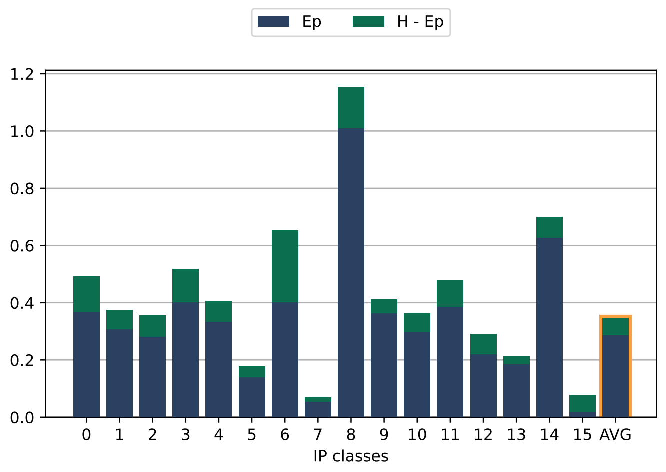

Fig. 5. IP class uncertainty.

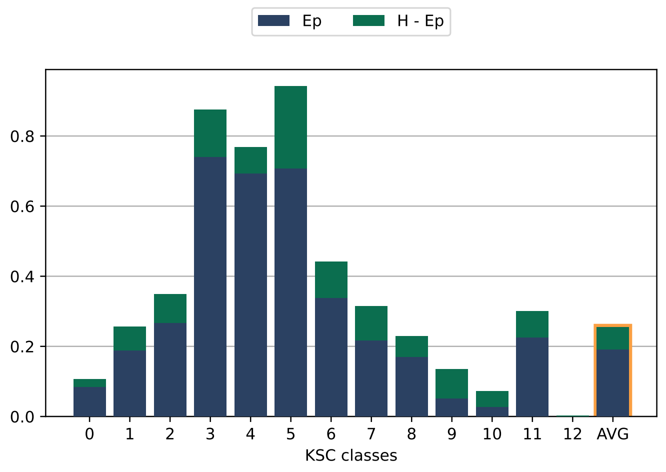

Fig. 6. KSC class uncertainty.

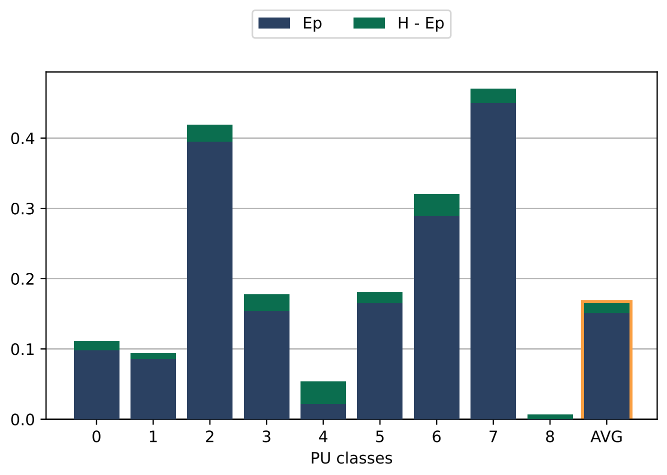

Fig. 7. PU class uncertainty.

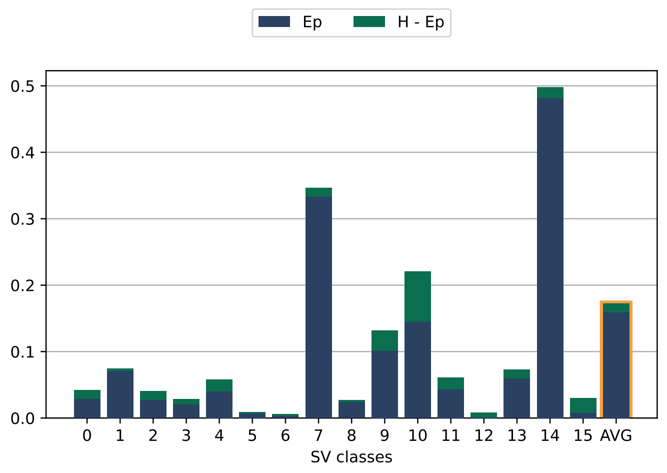

Fig. 8. SV class uncertainty.

intensive analysis of a particular image, their classes and characteristics is out of the scope of this paper, but here we can see how bayesian networks can help to identify problems in the dataset itself.

C. *Uncertainty maps*

A very interesting use of the information given by bayesian networks is the possibility of studying the uncertainty maps of the entire image. That give us very valuable information about how well our labelled data represents the entire scene, i.e., our prediction capabilities across the entire scene. Figures 9 to 13 show for each dataset the graphical representation of the averaged bayesian prediction for the entire scene, the uncertainty map and the ground truth to compare with the labelled classes. For better visualisation, the colours codification for all the images is shown in Table IV.

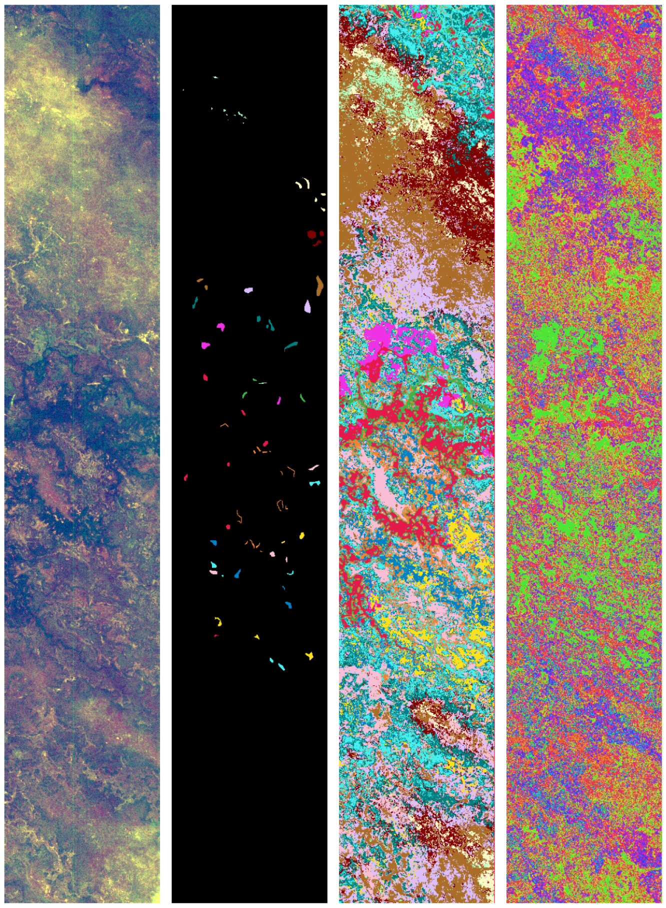
Fig. 9. From left to right: BO ground truth, prediction and uncertainty maps.

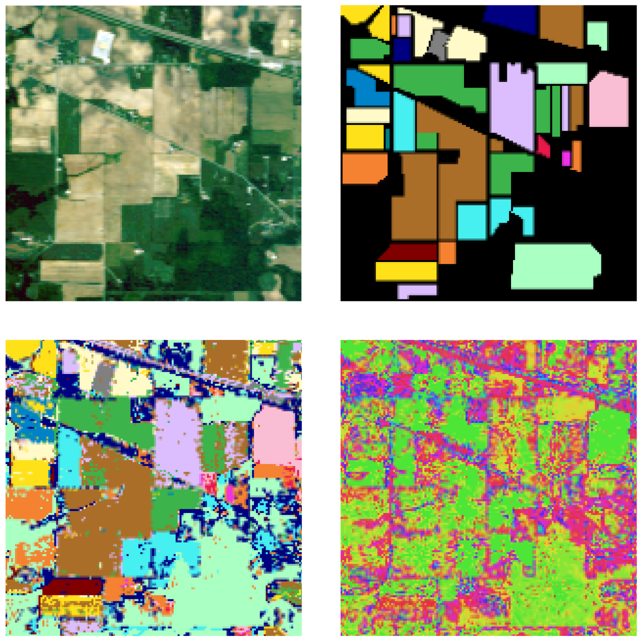
Fig. 10. From left to right: IP ground truth, prediction and uncertainty maps.

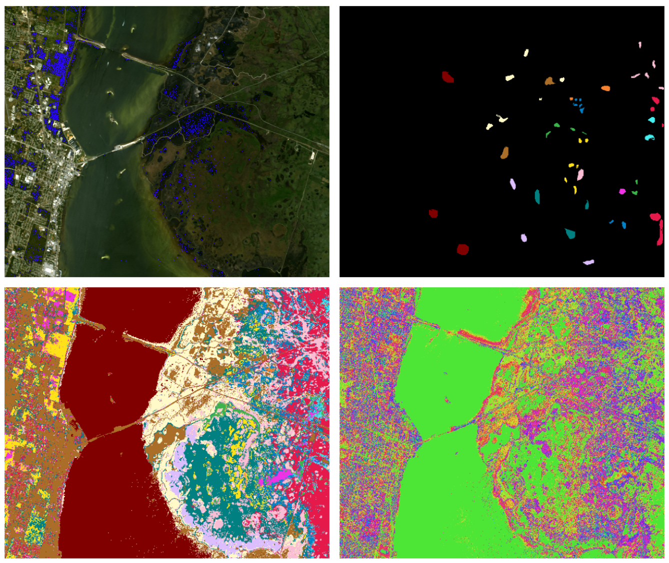
Fig. 11. From left to right: KSC ground truth, prediction and uncertainty maps.

TABLE IV
COLOURS CODIFICATION.

Classes		Ranges of uncertainty	
class 0	🟥	0.0 - 0.1	🟩
class 1	🟩	0.1 - 0.2	🟩
class 2	🟨	0.2 - 0.3	🟩
class 3	🟦	0.3 - 0.4	🟨
class 4	🟧	0.4 - 0.5	🟧
class 5	🟦	0.5 - 0.6	🟧
class 6	🟪	0.6 - 0.7	🟥
class 7	🟥	0.7 - 0.8	🟥
class 8	🟩	0.8 - 0.9	🟪
class 9	🟪	0.9 - 1.0	🟪
class 10	🟫	1.0 - 1.1	🟪
class 11	⬜	1.1 - 1.2	🟦
class 12	🟥	1.2 - 1.3	🟦
class 13	🟩	1.3 - 1.4	🟦
class 14	🟦	1.4 - 1.5	🟦
class 15	⬜		

One of the first things to notice looking at the uncertainty maps is that borders and limits are well defined an differentiated from the rest of the scene. That is expected and perfectly understandable, as this borders can imply zones with mixed classes, due to the great size of the surface that each pixel represents, or even paths or other adjacent non-labelled elements. This not only gives information about the terrain, but, being an expected behaviour, serves as a proof of the capability of this uncertainty metrics to show us whether the network is being capable of recognise some pixels or not.

Comparing with the ground truth, we can also appreciate how some of the classes are easier to recognise for the network than others. A clear example is shown in Figure 13. In SV,

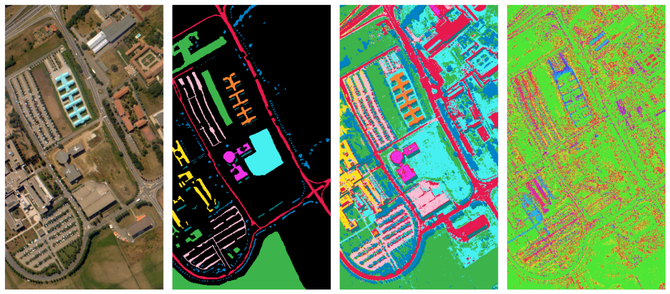
Fig. 12. From left to right: PU ground truth, prediction and uncertainty maps.

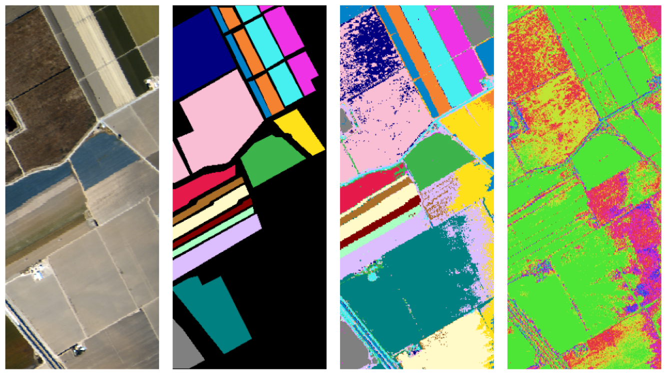
Fig. 13. From left to right: SV ground truth, prediction and uncertainty maps.

for most of the classes the accuracy is very high and the uncertainty very low. However, as previously mentioned, the model has problems in identifying class 14. As can be seen in the prediction map, it often confuses it with class 7. In these situations the uncertainty metric warns us by increasing its value as can be seen in the uncertainty map. It is also interesting to analyse the results of the non-labelled regions. In some cases, the model clearly identifies that they belong to one of the trained classes, whereas for other regions, it provides a prediction with a high value of uncertainty, indicating that its output is not reliable. This is a useful feature in order to use the model in real-world applications, since it may receive inputs that do not belong to any of the the trained categories.

D. Mixed classes

As a proof of concept, we are going to perform two particular tests with our datasets, the first one will be to train new networks after mixing the pixels of two classes on each training dataset in order to measure its effect on the aleatoric uncertainty for these two classes. The second one will be to feed the initial BNNs with data with an increasing level of random noise and to analyse analyse its effect on the uncertainty metrics. For the first experiment we selected two classes with a similar number of pixels for each image, shown in Table V, and we aleatory mixed their labels in the training set. For example in BO, half of the 269 pixels of class 4 and half of the 269 pixels of class 5 were selected for training, but before training their labels were randomly mixed up in such a way that half of the training pixels labelled with a 4 belonged to class 5 and vice versa. Since the categories are completely mixed, the model will not be able to distinguish between them. The objective of the experiment is to test if the model identifies this problem by increasing the aleatoric uncertainty.

TABLE V
SELECTED CLASSES AND NUMBER OF PIXELS.

	First class		Second class	
Image	Class number	Pixels	Class number	Pixels
BO	4	269	5	269
IP	2	830	5	730
KSC	8	520	11	503
PU	3	3064	7	3682
SV	1	3726	6	3579

Table VI shows the values of the aleatoric uncertainty of the BNN trainings with the mixed data ($Ep\ mixed$) compared to the same network trained with the initial data (Ep). As it was expected, the increase in the aleatoric uncertainty of the modified classes is very important. This behaviour tell us about how the expected entropy values of the different classes can be used to analyse the datasets characteristics and determine a possible lack of information for some of them.

TABLE VI
MIXED CLASSES ALEATORIC UNCERTAINTY (EP).

	First class		Second class		All classes average	
Image	Ep	Ep mixed	Ep	Ep mixed	Ep	Ep mixed
BO	0.23	0.81	0.53	0.89	0.19	0.28
IP	0.28	0.98	0.14	0.83	0.29	0.50
KSC	0.17	0.93	0.23	0.92	0.19	0.41
PU	0.15	0.83	0.45	1.02	0.15	0.29
SV	0.07	1.12	0.00	0.70	0.16	0.34

E. Uncertainty with Noise

Another interesting application of the bayesian networks is the detection of degradation in the quality of the input data, that can be generated due to an increase in the level of noise generated during the data acquisition or the communications. The idea is that if the level of noise in the inputs increases, the uncertainty metrics should be able to detect it.

For simulating this possible situations we progressively introduced random noise on the test bench and observe the uncertainty values. To generate the noise we added to each feature of each pixel a random 16 bits signed integer value multiplied by a noise factor, which we progressively incremented by 0.01 until the uncertainty converge to high values. The values of the uncertainty are shown in Figures 14 to 18, where each coloured line corresponds to each of the dataset classes, and the dashed lines corresponds to the average uncertainty value and the theoretical maximum uncertainty value of the dataset.

As we can observe, while we increment the noise factor the uncertainty value grows. It is not our intention here to try to determine a *measure* of the noise, as this will depend

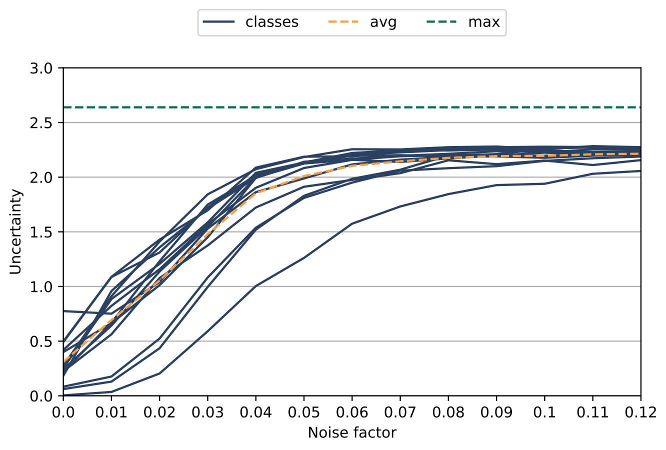

Fig. 14. BO uncertainty with noise.

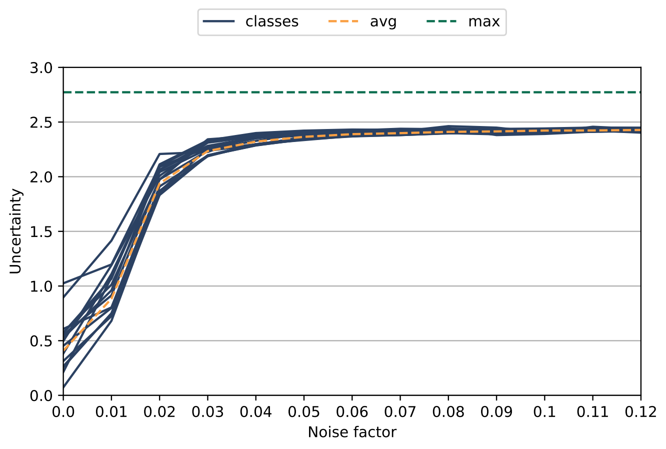

Fig. 15. IP uncertainty with noise.

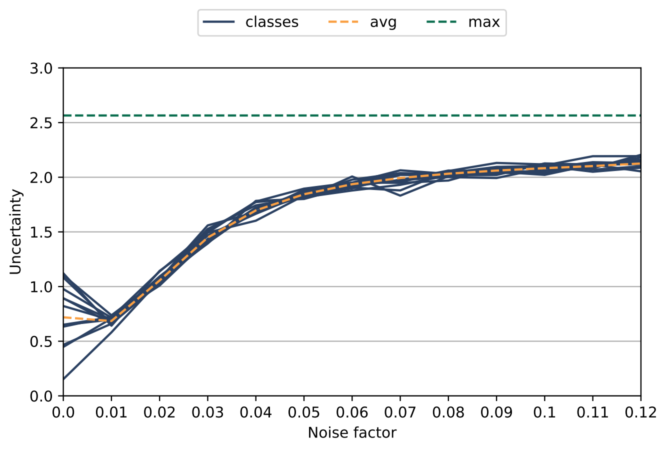

Fig. 16. KSC uncertainty with noise.

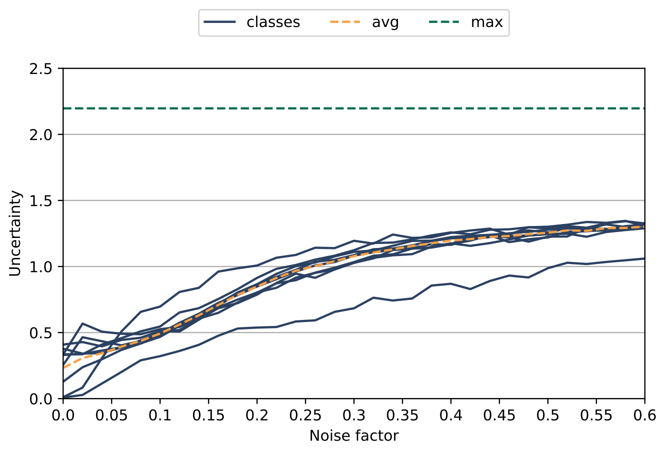

Fig. 17. PU uncertainty with noise.

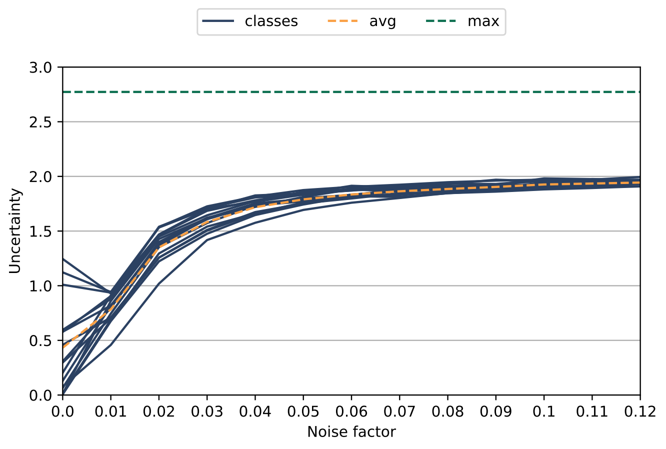

Fig. 18. SV uncertainty with noise.

on the noise source and the sensor and image characteristics, while we just used random generic noise added to the data. Analysing the figures, we can see that BO, IP, KSC and SV exhibit a similar behaviour and its uncertainty grows very fast from the beginning until it stabilises at around 0.04 noise factor. On the contrary, Pavia University (PU) dataset seems to be much more resistant to noise than the rest of datasets, with constant slower growth (note that the noise factor of Figure 17 has a different scale than the others). Being robust to noise depends on many factors such as data structure (number of observations, number of predictors, number of classes) and model complexity [27].

The results illustrate the utility of the uncertainty metrics to verify the quality of the input data. In the context of Remote Sensing, this is a very important feature, as the sensors and the data received are exposed to many external variations, such as climatic ones.

VI. Conclusion

In this work we wanted to evaluate how BNN models can help us to obtain calibrated and reliable solutions for a relevant problem: pixel classification of hyperspectral images. To this end, we trained a simple BNN for five of the most used hyperspectral images datasets and then explored the additional information that BNNs provide. First, we analysed the calibration of our model, which is an orthogonal task that can be done with any NN model, but is usually neglected. In the case of BNNs, using several stochastic passes contributes to achieve a good calibration. If network outputs are going to be used for decision-making, checking that the network is well calibrated should be a mandatory step. Next, we analysed the reliability of our models using the uncertainty metrics given by the probabilistic nature of the BNNs. The clear correlation between accuracy and uncertainty makes it possible to identify the outputs that provide a requested level of accuracy by selecting an appropriate uncertainty threshold. The output of the BNNs allows the uncertainty to be decomposed into two categories: aleatoric, caused by the data, and epistemic, caused by the model. Analysing the epistemic uncertainty we can verify that our model is appropriate for the given problem and is well trained, and with the aleatoric uncertainty we can analyse the quality of the dataset themselves, and also the quality of the data received during inference. A close exploration of the data sets showed that these datasets have important uncertainty differences between classes, which indicates that some of the classes are either miss-represented on the dataset, or they are too similar, or mislabelled. This can be helpful to identify that some data classes need more samples, or that two very similar categories should be unified, since their features are indistinguishable.

In order to further illustrate the added value of using BNNs, we performed two different experiments. First, we trained new BNNs after mixing the pixels of two classes on each training dataset. This leads to a large increase in uncertainty. Hence, the BNN was able to identify that there were problems with these classes. Second, we used the original BNNs and feed them with data with an increasing level of random noise. In the experiments, we observed that uncertainty works as an excellent measure of the input data quality during inference, giving us important information about possible noise factors or communication interference.

After these experiments, we believe that the benefits of upgrading models from NNs to BNNs are very clear. BNNs provide uncertainty metrics that can be used to identify problems in their outputs, or to evaluate the quality of the training data. The tools for designing BNNs have improved a lot in recent years, and some of the most widely used design environments, such as Tensorflow or Pytorch, include them. On the downside, the models used need twice as many parameters as an equivalent NN model. This can make it difficult to train very large models. This issue is interesting, and we would like to study it in the future, but in any case, we believe it is better to have a smaller but very reliable model, than a very large deep model that slightly improves the results, but does not provide uncertainty metrics.

VII. Acknowledgements

All authors acknowledge support from grants (1) PID2019-105660RB-C21 / AEI / 10.13039/501100011033 from Agencia Estatal de Investigación (AEI) and European Regional Development Fund (ERDF), and (2) gaZ: T5820R research group from Dept. of Science, University and Knowledge Society, Government of Aragón. And also from the Government of Aragón through the European Social Fund "Construyendo Europa desde Aragón" of the European Union. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

References

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, p. 436–444, 2015. [Online]. Available: https://doi.org/10.1038/nature14539

[2] A. Alcolea, M. E. Paoletti, J. M. Haut, J. Resano, and A. Plaza, "Inference in supervised spectral classifiers for on-board hyperspectral imaging: An overview," *Remote Sensing*, vol. 12, no. 3, 2020. [Online]. Available: https://www.mdpi.com/2072-4292/12/3/534

[3] S. Li, W. Song, L. Fang, Y. Chen, P. Ghamisi, and J. A. Benediktsson, "Deep learning for hyperspectral image classification: An overview," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 57, no. 9, pp. 6690–6709, 2019.

[4] D. Hong, L. Gao, J. Yao, B. Zhang, A. Plaza, and J. Chanussot, "Graph convolutional networks for hyperspectral image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 59, no. 7, pp. 5966–5978, 2021.

[5] M. Paoletti, J. Haut, J. Plaza, and A. Plaza, "A new deep convolutional neural network for fast hyperspectral image classification," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 145, pp. 120–147, 2018, deep Learning RS Data. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0924271617303660

[6] R. Schwartz, J. Dodge, N. A. Smith, and O. Etzioni, "Green ai," *Communications of the ACM*, vol. 63, no. 12, pp. 54–63, 2020.

[7] A. Alcolea and J. Resano, "Bnn for hyperspectral datasets analysis," https://github.com/universidad-zaragoza/BNN_for_hyperspectral_datasets_analysis, 2022, accessed on February 2022.

[8] J. M. Haut, A. Alcolea, M. E. Paoletti, J. Plaza, J. Resano, and A. Plaza, "Gpu-friendly neural networks for remote sensing scene classification," *IEEE Geoscience and Remote Sensing Letters*, vol. 19, pp. 1–5, 2022.

[9] Z. Ghahramani, "Bayesian non-parametrics and the probabilistic approach to modelling," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 371, no. 1984, p. 20110553, 2013.

[10] J. M. Haut, M. E. Paoletti, J. Plaza, J. Li, and A. Plaza, "Active learning with convolutional neural networks for hyperspectral image classification using a new bayesian approach," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 56, no. 11, pp. 6440–6461, 2018.

[11] Y. Gal and Z. Ghahramani, "Bayesian convolutional neural networks with bernoulli approximate variational inference," *arXiv preprint arXiv:1506.02158*, 2015.

[12] Y. Zhang, L. Wu, H. Ren, L. Deng, and P. Zhang, "Retrieval of water quality parameters from hyperspectral images using hybrid bayesian probabilistic neural network," *Remote Sensing*, vol. 12, no. 10, 2020. [Online]. Available: https://www.mdpi.com/2072-4292/12/10/1567

[13] U. Bhatt, J. Antorán, Y. Zhang, Q. V. Liao, P. Sattigeri, R. Fogliato, G. Melançon, R. Krishnan, J. Stanley, O. Tickoo, L. Nachman, R. Chunara, M. Srikumar, A. Weller, and A. Xiang, "Uncertainty as a form of transparency: Measuring, communicating, and using uncertainty," in *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*, ser. AIES '21. Association for Computing Machinery, 2021, p. 401–413. [Online]. Available: https://doi.org/10.1145/3461702.3462571

[14] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," in *Proceedings of the 34th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, D. Precup and Y. W. Teh, Eds., vol. 70. PMLR, 06–11 Aug 2017, pp. 1321–1330. [Online]. Available: https://proceedings.mlr.press/v70/guo17a.html

[15] B. Song, S. Sunny, S. Li, K. Gurushanth, P. Mendonca, N. Mukhia, S. Patrick, S. Gurudath, S. Raghavan, I. Tsusennaro, S. T. Leivon, T. Kolur, V. Shetty, V. R. Bushan, R. Ramesh, T. Peterson, V. Pillai, P. Wilder-Smith, A. Sigamani, A. Suresh, moni Abraham Kuriakose, P. Birur, and R. Liang, "Bayesian deep learning for reliable oral cancer image classification," *Biomed. Opt. Express*, vol. 12, no. 10, pp. 6422–6430, Oct 2021. [Online]. Available: http://opg.optica.org/boe/abstract.cfm?URI=boe-12-10-6422

[16] M. Sensoy, L. Kaplan, and M. Kandemir, "Evidential deep learning to quantify classification uncertainty," *arXiv preprint arXiv:1806.01768*, 2018.

[17] J. J. Senecal, J. W. Sheppard, and J. A. Shaw, "Efficient convolutional neural networks for multi-spectral image classification," in *2019 International Joint Conference on Neural Networks (IJCNN)*, 2019, pp. 1–8.

[18] R. M. Neal, *Bayesian Learning for Neural Networks.* Berlin, Heidelberg: Springer-Verlag, 1996.

[19] Z. Ghahramani, "Probabilistic machine learning and artificial intelligence," *Nature*, vol. 521, pp. 452–459, 05 2015.

[20] J. Antorán Cabiscol, "Understanding uncertainty in bayesian neural networks," *Master of Philosophy (University of Cambridge)*, 2019.

[21] GIC, "Hyperspectral remote sensing scenes, grupo de inteligencia computacional de la universidad del país vasco," [online] http://www.ehu.eus/ccwintco/index.php/Hyperspectral_Remote_Sensing_Scenes, accessed January 2022.

[22] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[23] TensorFlow, "TensorFlow denseflipout layer documentation," https://www.tensorflow.org/probability/api_docs/python/tfp/layers/DenseFlipout, [Online; accessed August 2021].

[24] Y. Wen, P. Vicol, J. Ba, D. Tran, and R. Grosse, "Flipout: Efficient pseudo-independent weight perturbations on mini-batches," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=rJNpifWAb

[25] M. P. Naeini, G. F. Cooper, and M. Hauskrecht, "Obtaining well calibrated probabilities using bayesian binning," in *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, ser. AAAI'15. AAAI Press, 2015, p. 2901–2907.

[26] A. Niculescu-Mizil and R. Caruana, "Predicting good probabilities with supervised learning," in *ICML 2005 - Proceedings of the 22nd International Conference on Machine Learning*, 01 2005, pp. 625–632.

[27] A. D. Rocha, T. A. Groen, A. K. Skidmore, R. Darvishzadeh, and L. Willemen, "Machine learning using hyperspectral data inaccurately predicts plant traits under spatial dependency," *Remote Sensing*, vol. 10, no. 8, 2018. [Online]. Available: https://www.mdpi.com/2072-4292/10/8/1263

Adrián Alcolea graduated in Social Work from the University of Zaragoza, Spain (2010). He obtained his master degree in International Peace, Conflict and Development Studies from the University Jaume I of Castellón, Spain (2012). Then he graduated in Computer Engineering from the University of Zaragoza, Spain (2017). He is currently a PhD student at the Computer Architecture Research Group of the University of Zaragoza, where he is working on the development of hardware support for DNNs, mainly focused on the development of efficient accelerators for Machine Learning techniques on Embedded Systems.

Javier Resano received the Bachelor Degree in Physics in 1997, a Master Degree in Computer Science in 1999, and the PhD degree in Computer architecture in 2005 at the Universidad Complutense of Madrid, Spain. He worked eight years as Assistant and Associate Professor at this University, and collaborated with the Interuniversity Microelectronics Center (IMEC) in Leuven, Belgium. Currently he is Associate Professor at the Computer Eng. Department of the Universidad de Zaragoza (Spain), and a member of the Gaz research group (Computer Architecture group from Universidad of Zaragoza) and the Aragón Institute for Engineering Research (I3A). His research has been focused in hardware/-software co-design, task scheduling techniques for multi-processor systems, Digital Logic design using FPGAs, dynamically reconfigurable hardware and efficient accelerators for machine learning and remote sensing.