A Deeper Look into Aleatoric and Epistemic Uncertainty Disentanglement

Matias Valdenegro-Toro
Department of AI, Bernoulli Institute
University of Groningen
m.a.valdenegro.toro@rug.nl

Daniel Saromo Mori
Artificial Intelligence Research Group
Pontifical Catholic University of Peru
daniel.saromo@pucp.pe

Abstract

Neural networks are ubiquitous in many tasks, but trusting their predictions is an open issue. Uncertainty quantification is required for many applications, and disentangled aleatoric and epistemic uncertainties are best. In this paper, we generalize methods to produce disentangled uncertainties to work with different uncertainty quantification methods, and evaluate their capability to produce disentangled uncertainties. Our results show that: there is an interaction between learning aleatoric and epistemic uncertainty, which is unexpected and violates assumptions on aleatoric uncertainty, some methods like Flipout produce zero epistemic uncertainty, aleatoric uncertainty is unreliable in the out-of-distribution setting, and Ensembles provide overall the best disentangling quality. We also explore the error produced by the number of samples hyper-parameter in the sampling softmax function, recommending $N > 100$ samples. We expect that our formulation and results help practitioners and researchers choose uncertainty methods and expand the use of disentangled uncertainties, as well as motivate additional research into this topic.

1. Introduction

Neural networks are state of the art for many tasks [2], ranging from Computer Vision [13] to Robotics [4], Natural Language Processing [5], and some Medical applications [2]. Use cases involving human subjects usually require some safety constraints and, in general, this means a model should produce reasonable estimates of its confidence or uncertainty when making predictions.

There are two kinds of uncertainty [8,12,14]: aleatoric or data uncertainty, and epistemic or model uncertainty. These uncertainties are usually combined and predicted as a single value, called predictive uncertainty [8]. Recovering the two components of uncertainty is helpful for certain applications. For example, in active learning, epistemic uncertainty can guide the selection of samples to label, but aleatoric uncertainty should be ignored. In robot perception (segmentation, object detection), it is helpful to separate data uncertainty from model uncertainty, for purposes of out-of-distribution detection, and to ignore outliers and noise. Figure 1 shows an example of disentangled uncertainties for a toy regression example.

There are methods to disentangle aleatoric and epistemic uncertainty for some machine learning models [6,7,20]. For deep neural networks, Kendall and Gal. [14] define a general disentangling model, but it is mostly defined for a base model using MC-Dropout [9]. In this paper, we generalize this formulation to allow disentanglement across multiple methods of uncertainty estimation (Like Ensembles, Flipout, etc).

We make an experimental comparison between different uncertainty quantification methods relative to their capacity for disentangling aleatoric and epistemic uncertainty. We tested those techniques in regression and classification tasks (on the FER+ dataset), and explore the interaction between both sources of uncertainty.

Overall, we find that for the purpose of disentangling, aleatoric and epistemic uncertainty do interact, which is unexpected, as only epistemic uncertainty should interact with the model, and not aleatoric uncertainty. In particular with Flipout, outputting only aleatoric uncertainty and zero epistemic uncertainty is helpful for certain applications.
temic uncertainty, even for out of distribution cases, which we believe is an anomaly. We also find that aleatoric uncertainty estimation is unreliable in out-of-distribution settings, particularly for regression, with constant aleatoric variances being output by a model. Our results show that Ensembles have the best uncertainty and disentangling behavior for both classification and regression, and the β-NLL loss [19] improves both aleatoric and epistemic uncertainty quantification, while Seitzer et al. had explored only its use for aleatoric uncertainty.

The contributions of this paper are a generalization of methods to disentangle aleatoric and epistemic uncertainty produced by a machine learning model across different uncertainty quantification methods, which were originally proposed by Kendall and Gal [14] but only for MC-Dropout; and a comparison between dropout, dropconnect, ensembles, and flipout, about their disentangling quality across a regression and classification tasks. We also explore setting the number of samples in the sampling softmax function, recommending the use $N = 100$ samples to prevent incorrect classification due to approximation error.

2. Related Work

As we mentioned before, separating the total uncertainty into its epistemic and aleatoric components is necessary for specific applications. There are some approaches for achieving this process [12]. Depeweg et al. [7] presented a method for measuring the total and aleatoric uncertainties, and then they calculated the epistemic element by subtracting those values. Nevertheless, they tested their proposal only in regression and reinforcement learning tasks. Alternatively, Kendall and Gal [14] developed another approach for independently calculating both uncertainty components using MC-Dropout. The authors tested their method at regression and classification problems. In this work we expand this disentanglement method to consider other uncertainty quantification methods.

On the other hand, we cannot know the exact probability distribution of the inputs that will be given to the trained model. Hence, quantify its uncertainty, we pass it samples from the training dataset, i.e. following the Empirical Risk Minimization (ERM) principle [21]. The most common sampling methods are: Monte Carlo Dropout [8, 9] (which turns off some activations at each sample passing to estimate the prediction uncertainty), Monte Carlo DropConnect [17] (which turns off weights instead of activations, and whose authors reported it to be capable of achieving better results than MC Dropout [17]), estimation with deep ensembles [16, 18] (which blends the predictions from networks with different weight initializations taken from the same probability distribution), Flipout-based variational inference [22] (which samples weight perturbations inside a mini-batch), and Markov Chain Monte Carlo [15] (which uses sequential drawings from a stochastic distribution to estimate the exact posterior); a review of these techniques can be found in [1].

3. Uncertainty Disentanglement

Many uncertainty quantification methods can be categorized into sampling-based and ensemble-based, where ensembling can be seen conceptually as a way of sampling. In this section we generalize the method proposed by Kendall and Gall [14].

3.1. Regression

Assume we have a model with uncertainty that outputs two quantities: the mean $\mu_i(x)$ and variance $\sigma^2_i(x)$, where $i \in [1, M]$ is an index for different samples or ensembles. For the purpose of uncertainty quantification, we usually sample weights from a weight distribution $\theta \sim p(\theta|x, y)$, which produce different predictions for $\mu$ and $\sigma^2$, that correspond to samples of the (approximate) predictive posterior distribution of the model.

These samples are usually combined into a single Gaussian mixture distribution $p(y|x)$ using:

$$p(y|x) \sim N(\mu_*(x), \sigma^2_*(x))$$ (1)

$$\mu_*(x) = M^{-1} \sum_i \mu_i(x)$$ (2)

$$\sigma^2_*(x) = M^{-1} \sum_i (\sigma^2_i(x) + \mu^2_i(x)) - \mu^2_*(x)$$ (3)
For the predictive variance $\sigma^2(x)$, this can be decomposed into aleatoric and epistemic uncertainty, by rewriting as:

$$
\sigma^2(x) = M^{-1} \sum \sigma_i^2(x) + M^{-1} \sum \mu_i^2(x) - \mu(x)^2
$$

$$
= \mathbb{E}_i[\sigma_i^2(x)] + \mathbb{E}_i[\mu_i^2(x)] - \mathbb{E}[\mu_i(x)]^2
$$

This derivation indicates that across forward pass samples, the mean of the variances represents aleatoric uncertainty, while the variance of the means corresponds to epistemic uncertainty. Also, this formulation can be derived by using the law of total variance.

The variance heads $\sigma^2(x)$ of a model can be trained using the Gaussian negative log-likelihood loss for a sample indexed by $n$ with input $x_n$ and label $y_n$:

$$
L_{NLL}(y_n, x_n) = \frac{\log \sigma_i^2(x_n)}{2} + \frac{(\mu_i(x_n) - y_n)^2}{2\sigma_i^2(x_n)}. \quad (4)
$$

This loss is also called variance attenuation. Nevertheless, this loss is known to have issues underestimating the variance head. Hence, an alternative called $\beta$-NLL [19] has been proposed to minimize these issues:

$$
L_{\beta-NLL}(y_n, x_n) = \text{stop}(\beta^2) L_{NLL}(y_n, x_n). \quad (5)
$$

Where stop($\cdot$) is the stop gradient operation, that prevents gradients from flowing through the operation inside the parenthesis. This loss makes the predicted variance act as a weight for each data point, putting more weight into larger variances. The parameter $\beta$ controls the strength of this weighting.

### 3.2. Classification

In classification problems, it is slightly more difficult to separately model aleatoric and epistemic uncertainty. Kendall and Gal [14] proposed to make a custom Softmax activation layer that models logits $z$ with uncertainty (Gaussian mean and variance), and uses sampling (with $N$ samples) to pass the Gaussian logit distribution $z$ through the softmax activation to produce $p(y|x)$. We call this function the sampling softmax function (Eq 6 and 7).

$$
\hat{z}_j \sim \mathcal{N}(\mu(x), \sigma^2(x))
$$

$$
p(y|x) = N^{-1} \sum_j \text{softmax}(\hat{z}_j) \quad j \in [1, N] \quad (7)
$$

Then at inference time, we again assume that the uncertainty method uses sampling through forwarding passes or ensembling a model on the $i$ axis (with $i \in [1, M]$). Then, aleatoric $\sigma^2_{Ae}$ and epistemic $\sigma^2_{Ep}$ uncertainty logits can be computed

$$
\sigma^2_{Ae}(x) = \mathbb{E}_i[\sigma_i^2(x)] \quad \sigma^2_{Ep}(x) = \text{Var}_i[\mu_i(x)]. \quad (8)
$$

Note that these are logits, not probabilities. Passing each corresponding logit through a softmax function can produce probabilities, from where entropy is a possible metric to obtain a scalar uncertainty measure:

$$
p_{Ae}(y|x) = \text{sampling_softmax}(\mu(x), \sigma^2_{Ae}(x))
$$

$$
H_{Ae}(y|x) = \text{entropy}(p_{Ae}(y|x))
$$

$$
p_{Ep}(y|x) = \text{sampling_softmax}(\mu(x), \sigma^2_{Ep}(x))
$$

$$
H_{Ep}(y|x) = \text{entropy}(p_{Ep}(y|x))
$$

Where $\mu(x) = M^{-1} \sum_i \mu_i(x)$ is the predictive mean and entropy is the standard Shannon entropy defined as $\text{entropy}(p) = - \sum_i p_i \log p_i$.

It should be noted that unlike the case of regression, in classification, disentangling uncertainties and transforming them into probabilities does not mean that epistemic and aleatoric probabilities or entropy will sum to the predictive probabilities or entropy. Only the logits can be summed to obtain predictive logits.

Figure 2 shows the behavior of the sampling softmax function with different logit distributions. The behavior is not so intuitive, particularly when the means of both logit distributions are the same. When the means are different, the logit variances play a more significant role and affect the final probabilities. Figure 3 shows the computational graph of this function.

### 3.3. Tuning the Number of Samples

Kendall and Gal [14] do not provide information on how the number of samples $N$ for the sampling softmax function should be selected. This parameter controls a trade-off between computational performance and error in the estimated probabilities. We evaluate this parameter by estimating the L2 error between $N = 100000$ and a variable value.
We use DropConnect layers with drop probability $p$, which also produces samples from the Bayesian posterior. DropConnect enables dropping weights at inference time, of activations, with a similar regularization effect. MC-DropConnect is conceptually similar to Dropout, randomly dropping weights to zero instead of activations, with a similar regularization effect. MC-DropConnect enables dropping weights at inference time, which also produces samples from the Bayesian posterior distribution [17]. We use DropConnect layers with drop probability $p = 0.10$.

Ensembles. Ensembles consist of training multiple copies of the same architecture (with different instances of a random weight initialization) and then combining their outputs, which usually produces a better model. Lakshminarayanan [16] demonstrated that ensembles also have good uncertainty quantification properties. We use an ensemble of $M = 5$ neural networks.

Flipout. Flipout-based variational inference is a popular method which models weights as an approximate Gaussian distribution [3], where the kernel and bias matrices are Gaussian distributed. This process generates a stochastic model. Flipout [22] is used to reduce the training process variance, improving learning stability and performance. We use Flipout in multiple layers with a disabled prior, and biases are scalars instead of distributions.

For evaluation we take $M = 20$ forward passes of each method. After that, we compute the mean of probabilities for classification, and the mean and standard deviation for regression, both across forward pass samples. The sampling softmax layer uses $N = 100$ samples for the classification task. As baseline, we also train a neural network without epistemic uncertainty quantification, which we denote as Classical NN. For the regression setting this network uses a mean and variance output heads, to be able to estimate aleatoric uncertainty.

4.2. Toy Regression

We first evaluate a simple regression problem, generating a dataset by sampling the following function:

$$f(x) = x \sin(x) + \epsilon_1 x + \epsilon_2$$  \hspace{1cm} (9)$$

Where $\epsilon_1, \epsilon_2 \sim \mathcal{N}(0, 0.3)$. This function has both homoscedastic ($\epsilon_2$) and heteroscedastic ($\epsilon_1$) aleatoric uncertainty, and a model has to estimate both. We produce 1000 samples for $x \in [0, 10]$ as a training set, and an out-of-distribution dataset is built with 200 samples for $x \in [10, 15]$.

We train models using the Gaussian NLL (Eq 4) and $\beta$-Gaussian NLL (Eq 5) with $\beta = 0.5$. These results are available in Figures 6 and 7. The training data that was used to
Figure 6. Comparison of four methods for disentangling uncertainty in a toy sinusoid problem with aleatoric uncertainty (homoscedastic and heteroscedastic), using the variance attenuation (NLL) loss. Dropout and DropConnect overestimate epistemic uncertainty on the training set, while ensembles and flipout have low uncertainty outside the training set.

Figure 7. Comparison of four uncertainty methods in disentangling uncertainty for a toy sinusoid problem with aleatoric uncertainty (homoscedastic and heteroscedastic), using the $\beta$-NLL loss. Dropout and DropConnect overestimate epistemic uncertainty on the training set, while ensembles has increased epistemic uncertainty outside the training set.

Our results show that aleatoric uncertainty estimation is unreliable in the out-of-distribution setting (values $x > 10$ in our experiments) for both losses. Ensembles and Flipout in particular produce constant aleatoric uncertainty in the out-of-distribution areas, indicating that they did not learn the general noise pattern in the data (where the noise variance $\epsilon_1$ would continue increasing), and it seems these methods only learned the term $\epsilon_2$ correctly. In contrast, it is interesting to see that the Classical NN did learn that aleatoric uncertainty is increasing even in the out-of-distribution areas.

For the Classical NN, Ensemble, and Flipout, the $\beta$-NLL loss allows us to obtain a similar variance as with the NLL loss, at both aleatoric and epistemic uncertainties. However, both losses generate similar epistemic uncertainty with the dropout method. Nevertheless, for Flipout, epistemic un-
The mean logit layer uses a linear activation, which produces the final output probabilities. This is for logits, which are given as input to the sampling softmax function described in Sec 3.2, first by using two fully connected layers. We implement the sampling softmax function as described in the supplementary, using a cross-entropy loss using probability distribution labels (not one-hot encoded) which represent a distribution over eight emotion classes (Neutral, Happiness, Surprise, Sadness, Anger, Disgust, Fear, Contempt). We implement the sampling softmax function as described in Sec 3.2, first by using two fully connected layers with $C = 8$ neurons, each predicting a mean and variance for logits, which are given as input to the sampling softmax, which produces the final output probabilities. This is shown in Figure 3. The mean logit layer uses a linear activation, while the logit variance logit layer uses a softplus activation to produce positive variances. The probabilities are supervised by the cross-entropy loss directly, and automatic differentiation is used to compute gradients through the sampling function.

We report test loss and accuracy in Table 1. For a qualitative evaluation, we select the top 5 images on the test set according to the entropy computed from the ground truth class probability distribution. Then we compute the disentangled aleatoric and epistemic logits using the method described in Sec 3.2. Finally, we transform them into probabilities using the mean and variance through the sampling softmax function. We present these results in Figures 8 to 12. Each figure shows the entropy of each probability distribution, as a measure of uncertainty. It should be noted that, unlike regression, the aleatoric and epistemic probabilities and entropy values do not add to the total predicted entropy.

Flipout seems to produce zero epistemic uncertainty for all the selected examples, which is odd but consistent with the toy regression example. We believe this is due to Flipout’s powerful variance reduction effect, which seems to affect epistemic uncertainty in particular, putting all uncertainty mass into aleatoric uncertainty.

In most cases, the aleatoric probabilities follow or are similar to the ground truth probabilities. There is a more significant disagreement between epistemic uncertainty across different uncertainty quantification methods, which is unexpected, as all methods use the same network topology, except for some differences in how specific UQ methods are applied to the architecture (e.g. replacing layers, adding Dropout layers). We believe these results show that adding UQ to a neural network has a more considerable impact on its epistemic uncertainty than the aleatoric one.

Table 1 presents accuracy and cross-entropy metrics in the FER+ dataset. The Ensembles approach is the evident best in terms of accuracy, but Dropout has the smallest loss, closely followed by ensembles. Accuracy is only evaluated for the highest probability prediction, and the ground truth probabilities in FER+ for many cases do not have a clear dominating class (for example in Figures 8, 9), while the cross-entropy loss measures a fit in terms of probability distributions.

### 4.4. Discussion

Our expectation is that aleatoric uncertainty should be independent of the uncertainty quantification method, while epistemic uncertainty should depend on the specific uncertainty method. This is because aleatoric uncertainty is associated to the data, so the model performs regression of the variance in the data, while epistemic is associated to the model itself, and the uncertainty quantification method interacts with the model directly. Both kinds of uncertainty should not interact with each other.

Our classification and regression results mainly show that the uncertainty quantification method does affect both aleatoric and epistemic uncertainty quantification, with some surprising results. As mentioned before, this is expected for epistemic but not for aleatoric uncertainty.

<table>
<thead>
<tr>
<th>UQ Method</th>
<th>Test Accuracy (%)</th>
<th>Loss</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>74.8 %</td>
<td>0.218</td>
</tr>
<tr>
<td>Dropout</td>
<td>76.7 %</td>
<td>0.124</td>
</tr>
<tr>
<td>DropConnect</td>
<td>74.3 %</td>
<td>0.167</td>
</tr>
<tr>
<td>Flipout</td>
<td>70.1 %</td>
<td>0.150</td>
</tr>
<tr>
<td>Ensemble</td>
<td>77.8 %</td>
<td>0.130</td>
</tr>
</tbody>
</table>

Table 1. Accuracy and loss metrics for FER+ facial emotion classification with uncertainty quantification.
In particular, Flipout seems to confuse aleatoric and epistemic uncertainty, producing almost zero epistemic uncertainty in both classification and regression settings. This behavior is unexpected and most likely incorrect, and we would not recommend its use if disentangling uncertainty is an application requirement.

Some uncertainty models like Dropout and DropConnect are heavily affected by the use of the $\beta$-NLL loss in a regression setting, which improves aleatoric uncertainty estimation, but seems to also affect epistemic uncertainty, mostly in a positive way. However, not all methods equally benefit. It is the case of Flipout, which seems to have smaller epistemic uncertainty for an incorrect prediction.
temic uncertainty, with little change to aleatoric uncertainty.

Overall the best combination of uncertainty method and loss seems to be Ensembles with the $\beta$-NLL loss for regression, and ensembles for classification, obtaining the best accuracy in our FER+ dataset experiments, and a good disentangling of aleatoric and epistemic uncertainty in our toy regression experiments.

We believe that these insights will help the community and practitioners select appropriate uncertainty quantification methods, and expand the use of disentanglement of aleatoric and epistemic uncertainty.

5. Conclusions and Future Work

In this paper, we generalized methods to disentangle aleatoric and epistemic uncertainty, and compared different uncertainty quantification methods in terms of their disentangling quality. Our results show the differences between using different uncertainty quantification methods. We illustrate how aleatoric and epistemic uncertainties interact with each other, which is unexpected and partially violates the definitions of each kind of uncertainty, specially for aleatoric uncertainty. In particular, Flipout seems to produce almost zero epistemic uncertainty, putting all mass into aleatoric uncertainty. Ensembles unsurprisingly seem to have the best disentangling quality, when trained using the $\beta$-NLL loss for regression, and with the sampling softmax function for classification.

We also explored the approximation error produced by the number of samples used in the sampling softmax function, and we show that at least $N = 100$ samples are required for a good approximation and to have a misclassification error close to zero.

We expect that our results can guide practitioners to select uncertainty quantification methods, include disentangling quality as an additional metric for future uncertainty quantification methods, and expand the use of disentangled uncertainty quantification methods.

We believe that our results show the need for additional research to understand how epistemic and aleatoric uncertainty interact when estimated using machine learning models, and ways to reduce interactions for aleatoric uncertainty.
References


