Important Uncertainties

Today, deep learning algorithms are able to learn powerful representations which can map high dimensional data to an array of outputs. However these mappings are often taken blindly and assumed to be accurate, which is not always the case.
If both these algorithms were able to assign a high level of uncertainty to their erroneous predictions.

Deep learning does not allow for uncertainty representation in regression settings for example, and deep learning classification models often give normalised score vectors, which do not necessarily capture model uncertainty.

In Bayesian modeling, there are two main types of uncertainty one can model.

1) **Aleatoric uncertainty** captures **noise** inherent in the observations. This could be for example sensor noise or motion noise, resulting in uncertainty which cannot be reduced even if more data were to be collected.

2) On the other hand, **epistemic uncertainty** accounts for **uncertainty in the model parameters** – uncertainty which captures our ignorance about which model generated our collected data. This uncertainty can be explained away given enough data, and is often referred to as model uncertainty

Bayesian deep learning models typically form uncertainty estimates by either placing distributions over model weights, or by learning a direct mapping to probabilistic outputs.

-> Epistemic uncertainty is modeled by placing a prior distribution over a model's weights.

-> Aleatoric uncertainty on the other hand is modeled by placing a distribution over the output of the model

---

Aleatoric uncertainty

Regression

To capture aleatoric uncertainty in regression, we would have to tune the observation noise parameter σ.

Homoscedastic regression assumes constant observation noise σ for every input point x. Heteroscedastic regression, on the other hand, assumes that observation noise can vary with input x. In non-Bayesian neural networks, this observation noise parameter is often fixed as part of the model's weight decay, and ignored.

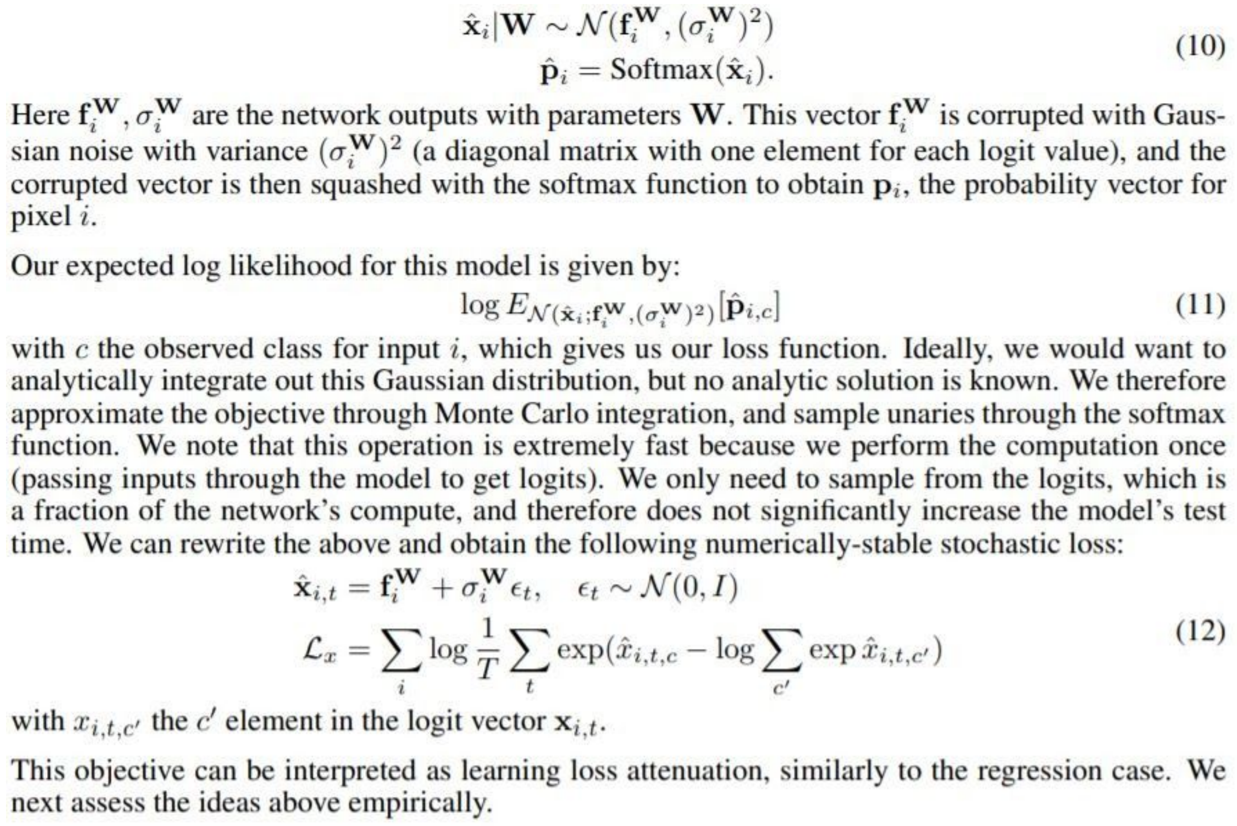
Firstly, we can model Heteroscedastic aleatoric uncertainty just by changing our loss functions. Because this uncertainty is a function of the input data, we can learn to predict it using a deterministic mapping from inputs to model outputs. For regression tasks, we typically train with something like a Euclidean/L2 loss: Loss= ||y-y'||2. To learn a Heteroscedastic uncertainty model, we simply can replace the loss function with the following:

$$Loss = \frac{||y - \hat{y}||_2}{2\sigma^2} + \frac{1}{2}\log\sigma^2$$

where the **model predicts a mean-(y hat) variance-σ2**. As you can see from this equation, if the model predicts something very wrong, then it will be encouraged to attenuate the residual term, by increasing uncertainty σ2. However, the log(σ2) prevents the uncertainty term growing infinitely large. This can be thought of as learned loss attenuation.

Classification Task

For this we adapt the standard classification model to marginalise over intermediate heteroscedastic regression uncertainty placed over the logit space.



$$\begin{aligned} \hat{\mathbf{x}}_i|\mathbf{W} &\sim \mathcal{N}(\mathbf{f}_i^{\mathbf{W}}, (\sigma_i^{\mathbf{W}})^2) \\ \hat{\mathbf{p}}_i &= \text{Softmax}(\hat{\mathbf{x}}_i). \end{aligned} \tag{10}$$

Here $\mathbf{f}_i^{\mathbf{W}}, \sigma_i^{\mathbf{W}}$ are the network outputs with parameters $\mathbf{W}$. This vector $\mathbf{f}_i^{\mathbf{W}}$ is corrupted with Gaussian noise with variance $(\sigma_i^{\mathbf{W}})^2$ (a diagonal matrix with one element for each logit value), and the corrupted vector is then squashed with the softmax function to obtain $\mathbf{p}_i$, the probability vector for pixel i.

Our expected log likelihood for this model is given by:

$$\log E_{\mathcal{N}(\hat{\mathbf{x}}_i;\mathbf{f}_i^{\mathbf{W}},(\sigma_i^{\mathbf{W}})^2)}[\hat{\mathbf{p}}_{i,c}] \tag{11}$$

with c the observed class for input i, which gives us our loss function. Ideally, we would want to analytically integrate out this Gaussian distribution, but no analytic solution is known. We therefore approximate the objective through Monte Carlo integration, and sample unaries through the softmax function. We note that this operation is extremely fast because we perform the computation once (passing inputs through the model to get logits). We only need to sample from the logits, which is a fraction of the network's compute, and therefore does not significantly increase the model's test time. We can rewrite the above and obtain the following numerically-stable stochastic loss:

$$\begin{aligned} \hat{\mathbf{x}}_{i,t} &= \mathbf{f}_i^{\mathbf{W}} + \sigma_i^{\mathbf{W}}\epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0, I) \\ \mathcal{L}_x &= \sum_i \log \frac{1}{T}\sum_t \exp(\hat{x}_{i,t,c} - \log\sum_{c'} \exp\hat{x}_{i,t,c'}) \end{aligned} \tag{12}$$

with $x_{i,t,c'}$ the c' element in the logit vector $\mathbf{x}_{i,t}$.

This objective can be interpreted as learning loss attenuation, similarly to the regression case. We next assess the ideas above empirically.

Epistemic uncertainty

----> Read Bayesian inference pdf

For regression tasks we often define our likelihood as a Gaussian with mean given by the model output: $p(\mathbf{y}|\mathbf{f}^{\mathbf{W}}(\mathbf{x})) = \mathcal{N}(\mathbf{f}^{\mathbf{W}}(\mathbf{x}), \sigma^2)$, with an observation noise scalar σ. For classification, on the other hand, we often squash the model output through a softmax function, and sample from the resulting probability vector: $p(\mathbf{y}|\mathbf{f}^{\mathbf{W}}(\mathbf{x})) = \text{Softmax}(\mathbf{f}^{\mathbf{W}}(\mathbf{x}))$.

The loss function is called the negative log likelihood function:

$$\mathcal{L}(\theta, p) = -\frac{1}{N}\sum_{i=1}^{N} \log p(\mathbf{y}_i|\mathbf{f}^{\widehat{\mathbf{W}}_i}(\mathbf{x}_i)) + \frac{1-p}{2N}||\theta||^2$$

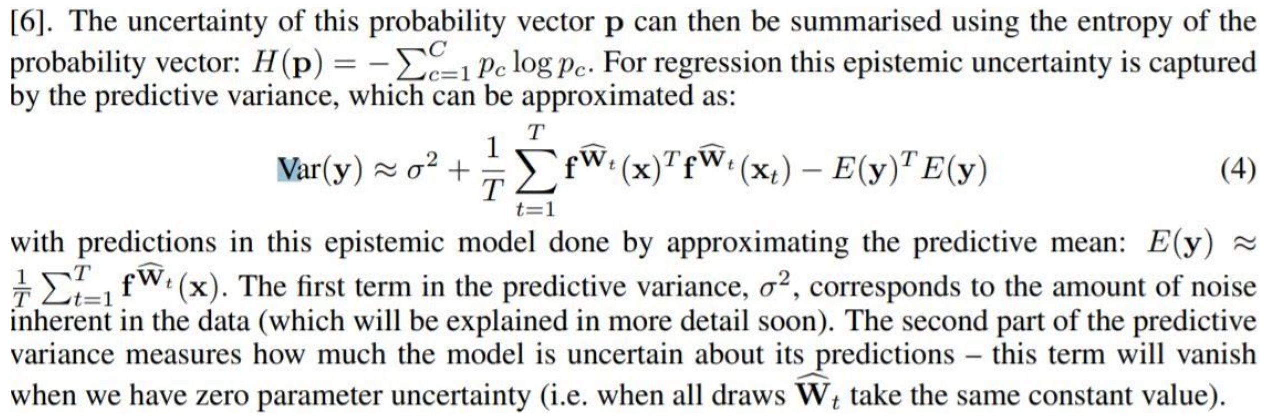
For regression it can be simplified as

$$-\log p(\mathbf{y}_i|\mathbf{f}^{\widehat{\mathbf{W}}_i}(\mathbf{x}_i)) \propto \frac{1}{2\sigma^2}||\mathbf{y}_i - \mathbf{f}^{\widehat{\mathbf{W}}_i}(\mathbf{x}_i)||^2 + \frac{1}{2}\log\sigma^2$$

for a Gaussian likelihood, with σ the model’s observation noise parameter – capturing how much noise we have in the outputs.

This uncertainty induces prediction uncertainty by marginalising over the (approximate) weights posterior distribution.

Calculating the Epistemic uncertainty quantitatively:



[6]. The uncertainty of this probability vector $\mathbf{p}$ can then be summarised using the entropy of the probability vector: $H(\mathbf{p}) = -\sum_{c=1}^{C} p_c \log p_c$. For regression this epistemic uncertainty is captured by the predictive variance, which can be approximated as:

$$\text{Var}(\mathbf{y}) \approx \sigma^2 + \frac{1}{T}\sum_{t=1}^{T} \mathbf{f}^{\widehat{\mathbf{W}}_t}(\mathbf{x})^T \mathbf{f}^{\widehat{\mathbf{W}}_t}(\mathbf{x}_t) - E(\mathbf{y})^T E(\mathbf{y}) \qquad (4)$$

with predictions in this epistemic model done by approximating the predictive mean: $E(\mathbf{y}) \approx \frac{1}{T}\sum_{t=1}^{T} \mathbf{f}^{\widehat{\mathbf{W}}_t}(\mathbf{x})$. The first term in the predictive variance, σ^2, corresponds to the amount of noise inherent in the data (which will be explained in more detail soon). The second part of the predictive variance measures how much the model is uncertain about its predictions – this term will vanish when we have zero parameter uncertainty (i.e. when all draws $\widehat{\mathbf{W}}_t$ take the same constant value).

Experimentations

In summary, we have demonstrated that our model can improve performance over non-Bayesian baselines by implicitly learning attenuation of systematic noise and difficult concepts.

We improve baseline performance by giving the model flexibility to estimate uncertainty and attenuate the loss.

What is IOU- https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/

In this section we evaluate our methods with **pixel-wise depth regression** and **semantic segmentation**. An analysis of these results is given in the following section. To show the robustness of our learned loss attenuation – a side-effect of modeling uncertainty – we present results on an array of popular datasets, CamVid, Make3D, and NYUv2 Depth, where we set new state-of-the-art benchmarks. For the following experiments we use the DenseNet architecture [19] which has been adapted for dense prediction tasks by [20]. We use our own independent implementation of the architecture using TensorFlow [21] (which slightly outperforms the original authors' implementation on CamVid by 0.2%, see Table 1a). For all experiments we train with 224 × 224 crops of batch size 4, and then fine-tune on full-size images with a batch size of 1. We train with RMS-Prop with a constant learning rate of 0.001 and weight decay 10−4 . We compare the results of the Bayesian neural network models outlined in §3. We model epistemic uncertainty using Monte Carlo dropout (§2.1). The DenseNet architecture places dropout with p = 0.2 after each convolutional layer. Following [22], we use 50 Monte Carlo dropout samples. We model aleatoric uncertainty with MAP inference using loss functions (8) and (12 in the appendix), for regression and classification respectively (§2.2). However, we derive the loss function using a Laplacian prior, as opposed to the Gaussian prior used for the derivations in §3. This is because it results in a loss function which applies a L1 distance on the residuals. Typically, we find this to outperform L2 loss for regression tasks in vision. We model the benefit of combining both epistemic uncertainty as well as aleatoric uncertainty using our developments presented in §3.

CamVid	IoU
SegNet [28]	46.4
FCN-8 [29]	57.0
DeepLab-LFOV [24]	61.6
Bayesian SegNet [22]	63.1
Dilation8 [30]	65.3
Dilation8 + FSO [31]	66.1
DenseNet [20]	66.9
This work:	
DenseNet (Our Implementation)	67.1
+ Aleatoric Uncertainty	67.4
+ Epistemic Uncertainty	67.2
+ Aleatoric & Epistemic	**67.5**

(a) CamVid dataset for road scene segmentation.

NYUv2 40-class	Accuracy	IoU
SegNet [28]	66.1	23.6
FCN-8 [29]	61.8	31.6
Bayesian SegNet [22]	68.0	32.4
Eigen and Fergus [32]	65.6	34.1
This work:		
DeepLabLargeFOV	70.1	36.5
+ Aleatoric Uncertainty	70.4	37.1
+ Epistemic Uncertainty	70.2	36.7
+ Aleatoric & Epistemic	**70.6**	**37.3**

(b) NYUv2 40-class dataset for indoor scenes.

Table 1: **Semantic segmentation performance.** Modeling both aleatoric and epistemic uncertainty gives a notable improvement in segmentation accuracy over state of the art baselines.

Make3D	rel	rms	$\log_{10}$
Karsch et al. [33]	0.355	9.20	0.127
Liu et al. [34]	0.335	9.49	0.137
Li et al. [35]	0.278	7.19	0.092
Laina et al. [26]	0.176	4.46	0.072
This work:			
DenseNet Baseline	0.167	3.92	0.064
+ Aleatoric Uncertainty	**0.149**	3.93	**0.061**
+ Epistemic Uncertainty	0.162	**3.87**	0.064
+ Aleatoric & Epistemic	**0.149**	4.08	0.063

(a) Make3D depth dataset [25].

NYU v2 Depth	rel	rms	$\log_{10}$	δ_1	δ_2	δ_3
Karsch et al. [33]	0.374	1.12	0.134	-	-	-
Ladicky et al. [36]	-	-	-	54.2%	82.9%	91.4%
Liu et al. [34]	0.335	1.06	0.127	-	-	-
Li et al. [35]	0.232	0.821	0.094	62.1%	88.6%	96.8%
Eigen et al. [27]	0.215	0.907	-	61.1%	88.7%	97.1%
Eigen and Fergus [32]	0.158	0.641	-	76.9%	95.0%	98.8%
Laina et al. [26]	0.127	0.573	0.055	81.1%	95.3%	98.8%
This work:						
DenseNet Baseline	0.117	0.517	0.051	80.2%	95.1%	98.8%
+ Aleatoric Uncertainty	0.112	0.508	0.046	81.6%	95.8%	98.8%
+ Epistemic Uncertainty	0.114	0.512	0.049	81.1%	95.4%	98.8%
+ Aleatoric & Epistemic	**0.110**	**0.506**	**0.045**	**81.7%**	**95.9%**	**98.9%**

(b) NYUv2 depth dataset [23].

Table 2: **Monocular depth regression performance.** Comparison to previous approaches on depth regression dataset NYUv2 Depth. Modeling the combination of uncertainties improves accuracy.

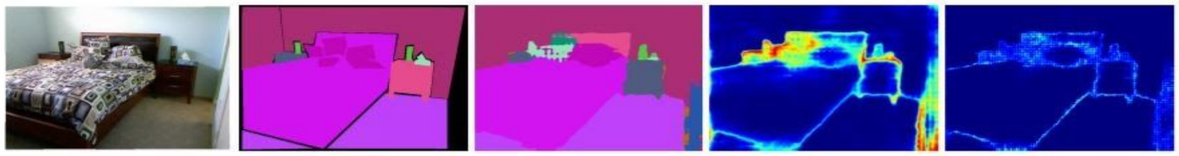


Figure 4: NYUv2 40-Class segmentation. From top-left: input image, ground truth, segmentation, aleatoric and epistemic uncertainty.

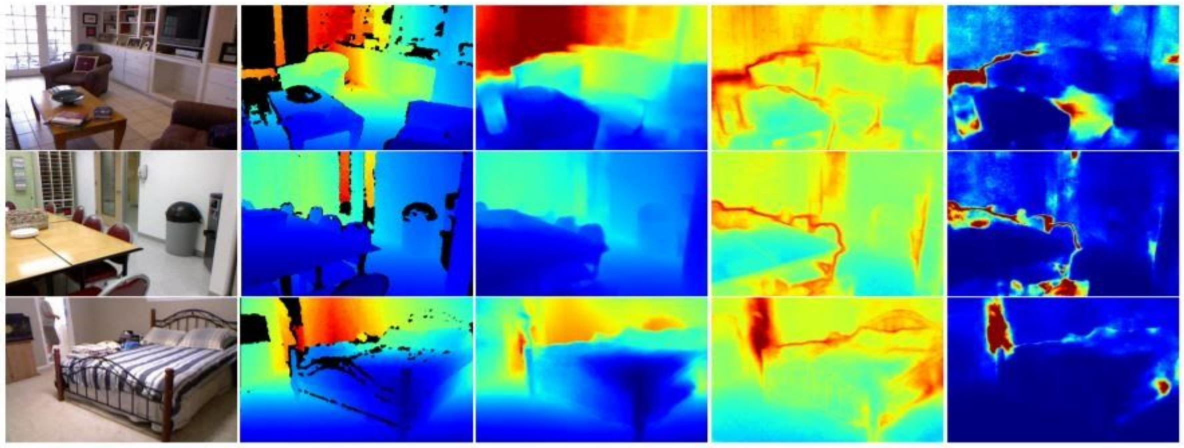


Figure 5: NYUv2 Depth results. From left: input image, ground truth, depth regression, aleatoric uncertainty, and epistemic uncertainty.

The implicit attenuation obtained from the **aleatoric loss provides a larger improvement** than the epistemic uncertainty model. However, the combination of both uncertainties improves performance even further. This shows that for this application it is more important to model aleatoric uncertainty, suggesting that **epistemic uncertainty can be mostly explained away in this large data setting.**

For example, in the qualitative results we observe that **aleatoric uncertainty is greater for large depths, reflective surfaces and occlusion boundaries in the image**. These are common failure modes of monocular depth algorithms. On the other hand, these qualitative results show that **epistemic uncertainty captures difficulties due to lack of data.**

Conclusion

We presented a novel Bayesian deep learning framework to learn a mapping to aleatoric uncertainty from the input data, which is composed on top of epistemic uncertainty models. We derived our framework for both regression and classification applications.

We showed that it is important to model aleatoric uncertainty for:

- Large data situations, where epistemic uncertainty is explained away,
- Real-time applications, because we can form aleatoric models without expensive Monte Carlo samples.

And epistemic uncertainty is important for:

- Safety-critical applications, because epistemic uncertainty is required to understand examples which are different from training data,
- Small datasets where the training data is sparse.

However aleatoric and epistemic uncertainty models are not mutually exclusive. We showed that the combination is able to achieve new state-of-the-art results on depth regression and semantic segmentation benchmarks.