What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision? A review

Nathaniel Cogneaux
Paris Dauphine-PSL
`nathaniel.cogneaux@dauphine.eu`

Abstract

Most deep learning models learn from given data and focus solely on improving performance. However, each model harbors its own uncertainties. In their paper, Alex Kendall and Yarin Gal identify two major types of uncertainty that can be modeled: aleatoric and epistemic. Aleatoric uncertainty is associated with noise inherent in the data itself, whereas epistemic uncertainty pertains to uncertainty within the model. With new Bayesian deep learning tools, it is now possible to model these uncertainties in computer vision tasks [2]. Kendall and Gal introduce a unified framework that combines input-dependent aleatoric uncertainty with epistemic uncertainty, leading to a new loss function. They demonstrate that this formulation, which can be interpreted as learned loss attenuation, enhances robustness to noisy data and achieves new state-of-the-art results in segmentation and depth regression benchmarks. I will try to engage with the paper critically and constructively, providing a comprehensive explanation of the theoretical methods employed and highlight its impact in today's research.

1 Introduction

Recalling the two types of uncertainties aforementioned: aleatoric uncertainty, which is intrinsic to the dataset and unaffected by additional data, is distinguished into homoscedastic (consistent across all inputs) and heteroscedastic (varying across different inputs due to factors like changing lighting conditions). On the other hand, epistemic uncertainty arises from the model's limitations. Extending the 2016 work by Gal and Ghahramani [2], this research advances our understanding of uncertainty in computer vision by integrating a Bayesian deep learning framework. The authors introduce a novel method that combines aleatoric and epistemic uncertainties in Bayesian deep learning to improve performance in computer vision tasks like depth estimation and semantic segmentation. The main idea is that, the variance of the output is also predicted by the model in addition to the output for aleatoric uncertainty while marginalizing the parameters for epistemic uncertainty. Methods to deal with each uncertainty already existed, but this is the first method to deal with both. The combination is valuable and this paper gives interesting analysis on the two uncertainties. Indeed, they provide insight on what the current deep learning methods can and cannot deal with. The paper concludes by claiming that aleatoric uncertainty is particularly important for large data situations and real-time applications, while epistemic uncertainty is particularly important for safety-critical applications and small datasets. The key outcomes of this work include:

- A novel classification approach that captures a nuanced understanding of both aleatoric and epistemic uncertainties.
- An enhancement in model performance by 1-3% over traditional non-Bayesian methods by reducing the impact of noisy data via explicit aleatoric uncertainty representation.

Preprint. Under review.

- An analysis of the trade-offs between aleatoric and epistemic uncertainties, examining their properties and implications for model performance and inference speed.
- The authors experimentally validate the claim that aleatoric uncertainty cannot be reduced with more data, and that only epistemic uncertainty increases for out-of-data inputs.

A quick remark: In this paper, the authors use epistemic uncertainty as a synonym for model uncertainty. Defining both, model uncertainty is the ignorance about the true physical model which creates the data; Epistemic uncertainty is supposed to capture everything which is unknown, but is deterministic and could in theory be known. So, numerical uncertainty is missing here. In large dimensional optimisation problems, it is completely unclear whether the optimisation algorithm has converged. Hence, this numerical error, i.e. the difference between the true minimum and the output of the optimiser, is a very significant source of error, which we are uncertain about, in particular when the computation has to be performed online, e.g. when the self-driving car learns while it's driving. Additionally, even rounding errors can add to the uncertainty of deep learning, as is e.g. discussed in this paper [3]. However, in the following I will mention epistemic uncertainty the way the authors meant it.

2 Traditional approaches & Main concepts

In 2017, the classical approaches to Bayesian deep learning typically captured either epistemic uncertainty or aleatoric uncertainty alone, depending on where the prior distribution is placed. Placing the prior over the model's weights captures epistemic uncertainty by indicating the variation in weights given the data. Conversely, placing the prior over the model's output captures aleatoric uncertainty.

2.1 Capturing epistemic uncertainty using Bayesian Neural Networks

Bayesian neural network (BNN) consists in putting a prior distribution over the weights with this idea of replacing the deterministic network's weight parameters with distributions over these parameters. In practice, the prior is a Gaussian: $W \sim \mathcal{N}(0, I)$. Denoting the random output of the BNN as $f^W(x)$ and the likelihood as $\mathbb{P}(y|f^W(x))$, then given the dataset $X = \{x_1, \dots, x_N\}, Y = \{y_1, \dots, y_N\}$ we use Bayesian inference to compute the posterior $\mathbb{P}(W|X, Y)$. Doing so, we capture the set of plausible model parameters, given the data, allowing us to quantify epistemic uncertainty. However, approximating the posterior over weights in these networks usually comes at a high computational cost. This is because the marginal probability $\mathbb{P}(Y|X)$, required to compute the posterior $\mathbb{P}(W|X, Y) = \frac{\mathbb{P}(Y|X,W)\mathbb{P}(W)}{\mathbb{P}(Y|X)}$, cannot be evaluated analytically.

Different approximations exist. Usually, in these approximate inference techniques, the posterior $\mathbb{P}(W|X, Y)$ is fitted with a simple distribution $q^*_\theta(W)$, parameterised by θ. This replaces the intractable problem of averaging over all weights in the BNN with an optimisation task, where we seek to optimise over the parameters of the simple distribution instead of optimising the original neural network's parameters.

Here, we consider Gal and Ghahramani's approach [2] which efficiently approximates the posterior with a Bernoulli variational distribution (instead of the usual Gaussian) through employing dropout in the network, see Figure 5. This is a realization that, the use of the regular dropout can be interpreted as a Bayesian approximation of a well-known probabilistic model: the Gaussian process. This inference is done by training a model with dropout before every weight layer, and by also performing dropout at test time to sample from the approximate posterior (stochastic forward passes, referred to as Monte Carlo dropout). So, instead of one prediction, we get many, one by each model. As illustrated in Figure 2, we can then average them or analyze their distributions providing mathematical grounds to reason about the model's uncertainty. The minimisation objective of this method is given by:

$$\mathcal{L}(\theta, p) = -\frac{1}{N}\sum_{i=1}^{N} \log \mathbb{P}(y_i|f_{\hat{W}_i}(x_i)) + \frac{1-p}{2N}||\theta||^2 \tag{1}$$

with N data points, dropout rate p, samples $\hat{W}_i \sim q^*_\theta(W)$ (Bernoulli this time), and θ the set of the simple distribution's parameters to be optimised (weight matrices in dropout's case).

For regression tasks we often define our likelihood as a Gaussian with mean given by the model output:

$$\mathbb{P}(y|f^W(x)) = \mathcal{N}(f^W(x), \sigma^2),$$

with an observation noise scalar σ capturing how much noise we have in the outputs. So:

$$-\log \mathbb{P}(y_i|f_{\widetilde{W}_i}(x_i)) \propto \frac{1}{2\sigma^2}||y_i - f_{\widetilde{W}_i}(x_i)||^2 + \frac{1}{2}\log\sigma^2 \tag{2}$$

The epistemic uncertainty is captured by the predictive variance, which can be approximated as:

$$\text{Var}(y) \approx \sigma^2 + \frac{1}{T}\sum_{t=1}^{T} f_{\hat{W}_t}(x)^T f_{\hat{W}_t}(x) - (\frac{1}{T}\sum_{t=1}^{T} f_{\hat{W}_t}(x))^T(\frac{1}{T}\sum_{t=1}^{T} f_{\hat{W}_t}(x))$$

The first term in the predictive variance, σ^2, corresponds to the amount of noise inherent in the data. 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

For classification tasks, on the other hand, we often squash the model output through a softmax function, and sample from the resulting probability vector:

$$\mathbb{P}(y|f^W(x)) = \text{Softmax}(f^W(x)).$$

This can be approximated using Monte Carlo integration as follows:

$$\mathbb{P}(y = c \mid x, X, Y) \approx \frac{1}{T}\sum_{t=1}^{T} \text{Softmax}(f_{\hat{W}_t}(x))$$

with T sampled masked model weights $\hat{W}_t \sim q_\theta^*(W)$, where $q_\theta(W)$ is the Dropout distribution. The uncertainty of this probability vector p can then be summarised using the entropy of the probability vector: $H(p) = -\sum_{c=1}^{C} p_c \log p_c$.

2.2 Capturing Heteroscedastic Aleatoric Uncertainty

Homoscedastic regression implies uniform observation noise σ across all inputs x, whereas heteroscedastic regression recognizes that this noise may change with x. In this work, we focus on the latter as it encompasses the former, making it more broadly applicable. Supposing the normality of residuals (important assumption) from our model f, which can be thought of as uncertainty about the data. One has:

$$\log \mathbb{P}(y|f(x)) = \log\left(\frac{1}{\sqrt{2\pi\sigma(x)}}\right) + \frac{\|y - f(x)\|^2}{2\sigma(x)^2}$$

In non-Bayesian neural networks, this observation noise parameter σ is often fixed as part of the model's weight decay, and ignored. However, when made data-dependent, it can be learned as a function of the data:

$$\mathcal{L}_{\text{NN}}(\theta) = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{1}{2\sigma(x_i)^2}\|y_i - f(x_i)\|^2 + \frac{1}{2}\log\sigma(x_i)^2\right) \tag{3}$$

The first term on the left side is the MSE term, and the second term acts as a regularization term for data uncertainty. This means preventing data uncertainty from growing meaninglessly. Consequently, samples with lower uncertainty have a higher impact on the loss and confusing samples have less impact on the learning process. In the paper this is referred to as **loss attenuation**. This way, the model is more robust to uncertainty in the data.

Note that here, unlike the above, variational inference is not performed over the weights, but instead they perform MAP inference – finding a single value for the model parameters θ. This approach does not capture epistemic model uncertainty.

3 Main Result: Combining Aleatoric and Epistemic Uncertainty

Now, the authors introduce their new models that examine aleatoric and epistemic uncertainties, either separately or combined. They highlight that aleatoric uncertainty can improve model robustness in regression tasks by acting as learned loss attenuation and adapt these concepts to classification tasks as well.

3.1 Combining Heteroscedastic Aleatoric Uncertainty and Epistemic Uncertainty

Looking at the heteroscedastic NN loss (3), it is enough to change this into a BNN, placing a distribution over its weights to quantify also the uncertainty of the model. To know this posterior distribution, they use the dropout variational inference methodology as mentioned in 2.1. Formally, we draw model weights from the approximate posterior $\hat{W} \sim q(W)$ to obtain a model output, this time composed of both predictive mean as well as predictive variance:

$$[\hat{y}, \hat{\sigma}^2] = f^{\hat{W}}(x) \tag{4}$$

where f, for vision tasks is a Bayesian convolutional neural network (BCNN) parametrised by model weights $\hat{W}$.

This induces a new minimisation objective given labeled output points x:

$$\mathcal{L}_{BNN}(\theta) = \frac{1}{D}\sum_i \left(\frac{1}{2}\hat{\sigma}_i^{-2}\|y_i - \hat{y}_i\|^2 + \frac{1}{2}\log\hat{\sigma}_i^2\right) \tag{5}$$

where D is the number of output pixels y_i corresponding to input image x, $\hat{\sigma}_i^2$ the BNN output for the predicted variance for pixel i. In cases like image-level regression tasks, D becomes 1, and for dense prediction tasks like semantic segmentation, D becomes the number of pixels.

As highlighted by (3), the loss function features **loss attenuation** through a dual-component structure, blending mean squared error (MSE) with a variance-based regularization. This design not only facilitates implicit variance learning but also strategically moderates the model's response to data uncertainty, bolstering robustness against noisy or uncertain data points.

In practice, to enhance numerical stability, we train the network to predict the log variance $s_i := \log\hat{\sigma}_i^2$, avoiding also a potential division by zero:

$$\mathcal{L}_{BNN}(\theta) = \frac{1}{D}\sum_i \left(\frac{1}{2}\exp(-s_i)\|\mathbf{y}_i - \hat{\mathbf{y}}_i\|^2 + \frac{1}{2}s_i\right). \tag{6}$$

Therefore, the method to obtain uncertainty in pixel(s) y is as follows:

$$\text{Var}(y) \approx \frac{1}{T}\sum_{t=1}^{T}\hat{y}_t^2 - \left(\frac{1}{T}\sum_{t=1}^{T}\hat{y}_t\right)^2 + \frac{1}{T}\sum_{t=1}^{T}\hat{\sigma}_t^2 \tag{7}$$

with $\{\hat{y}_t, \hat{\sigma}_t^2\}_{t=1}^{T}$ a set of T sampled outputs: $\hat{y}_t, \hat{\sigma}_t^2 = f^{\hat{W}_t}(x)$ for randomly masked weights $\hat{W}_t \sim q(W)$.

3.2 A novel approach for classification

Heteroscedastic neural networks (NNs) provide beneficial loss attenuation in regression and are adaptable to classification tasks, despite the inherent input-dependent uncertainty in classification. By marginalizing over heteroscedastic regression uncertainties within the logit space, they extend these concepts to classification, resulting in a novel adaptation named heteroscedastic classification NN.

Formally, after predicting a vector of unaries f_i for each pixel i, which when passed through a softmax operation, forms a probability vector p_i. They change the model by placing a Gaussian distribution over the unaries vector:

$$\hat{x}_i|W \sim \mathcal{N}(f_i^W, (\sigma_i^W)^2) \tag{8}$$

$$\hat{p}_i = \text{Softmax}(\hat{x}_i). \tag{9}$$

The vector f_i^W is corrupted with Gaussian noise with variance $(\sigma_i^W)^2$ (a diagonal matrix with one element for each logit value). Our expected log likelihood for this model is given by:

$$\log \mathbb{E}_{\mathcal{N}(\hat{x}_i; f_i^W, (\sigma_i^W)^2)}[\hat{p}_{i,c}] \tag{10}$$

with c the observed class for input i, which gives us our loss function. Here, the Gaussian distribution can be approximated by passing Monte-Carlo integration and finally Softmax. When this happens, the following equation appears:

$$\hat{x}_{i,t} = f_i^W + \sigma_i^W \varepsilon_t, \quad \varepsilon_t \sim \mathcal{N}(0, I) \tag{11}$$

$$\mathcal{L}_x = \sum_i \log \frac{1}{T} \sum_t \exp(\hat{x}_{i,t,c} - \log \sum_{c'} \exp \hat{x}_{i,t,c'}) \tag{12}$$

with $\hat{x}_{i,t,c'}$ the c' element in the logit vector $\hat{x}_{i,t}$. This new loss corresponds to the mean of the "SoftmaxCrossEntropy". This objective can be interpreted as learning loss attenuation, similarly to the regression case.

3.3 Going further

The reason why this paper is so interesting is because its methodology can be easily extended. For example, instead of just looking at depth regression or semantic segmentation, one can try doing both at the same time, that is called " Multi Task Scene Understanding Model " in [8] and applying the same loss methodology, one has:

$$\text{Loss} = \frac{L_{\text{regression 1}}}{\sigma_1^2} + \log \sigma_1^2 + \frac{L_{\text{regression 2}}}{\sigma_2^2} + \log \sigma_2^2 + \text{SoftmaxCrossEntropy}\left(\frac{y}{\sigma_3^2}\right)$$

Where the first term represents Depth Regression, the second Instance Regression and the third Semantic Segmentation.

4 Experiments

Then to show the robustness of their learned loss attenuation – a side-effect of modeling uncertainty – the authors present results on an array of popular datasets, CamVid, Make3D, and NYUv2 Depth, where they set new state-of-the-art benchmarks. And it appears indeed that modeling aleatoric and epistemic uncertainties enhance model performance, with the combination performing even better, by 1-3% over traditional non-Bayesian methods.

Training an autoencoder based on [7] to try to reconstruct Fashion MNIST dataset seems to already be enough to gather some interesting results, see the differences between aleatoric Figure 4, epistemic Figure 5 and combined uncertainties Figure 6.

Going further into the analysis using MNIST digits dataset we can observe that epistemic see Figure 10 and aleatoric see Figure 8 uncertainties seem to capture the same phenomenon when modeled separately which is the uncertainty related to contours reconstruction. However, modeling the two uncertainties together yields very different results. Indeed, the two uncertainties now seem to capture extremely complementary information when modeled together see Figure 12. And, weirdly enough, epistemic uncertainty is maximal on the background see Figure 13. The uncertainty related to digits contours is perhaps already captured by the aleatoric uncertainty branch and there is nothing left to be modeled but background uncertainty.

5 Conclusion

While Monte Carlo dropout is recognized as an effective method for modeling uncertainty, recent advancements have introduced more refined approaches to uncertainty quantification. These advancements encompass methods like Deep Evidential Learning, Deep Gaussian Processes, and various ensemble techniques including Deep Ensemble, Hyperparameter Ensembles, and Batch Ensemble. Additionally, the multi-input multi-output (MIMO) model, as introduced by Havasi et al. [6], along with DropConnect and Markov chain Monte Carlo for approximate inference, represents a significant evolution in the field. Despite the emergence of these methods, this paper continues to be widely cited, reflecting its foundational role in unifying approaches to learning under uncertainty within the domain of computer vision. The introduction of Bayesian Convolutional Neural Networks (CNNs), in particular, has been pivotal, credited with pioneering techniques to disentangle aleatoric and epistemic uncertainties in predictions, thus advancing different uncertainty quantification methods within machine learning models.

References

[1] A. Kendall and Y. Gal. What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision? In *Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS 2017)*, Long Beach, CA, USA, 2017.

[2] Y. Gal and Z. Ghahramani. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. arXiv:1506.02142v6 [stat.ML], Oct 2016.

[3] S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan. Deep Learning with Limited Numerical Precision. arxiv:1502.02551 In *Proceedings of the 32nd International Conference on Machine Learning (ICML 2015)*, Lille, France, 2015.

[4] M. Abdar, F. Pourpanah, S. Hussain, D. Rezazadegan, L. Liu, M. Ghavamzadeh, P. Fieguth, X. Cao, A. Khosravi, U. R. Acharya, V. Makarenkov, S. Nahavandi, "A Review of Uncertainty Quantification in Deep Learning: Techniques, Applications, and Challenges," arXiv:2011.06225v4 [cs.LG], 6 Jan 2021.

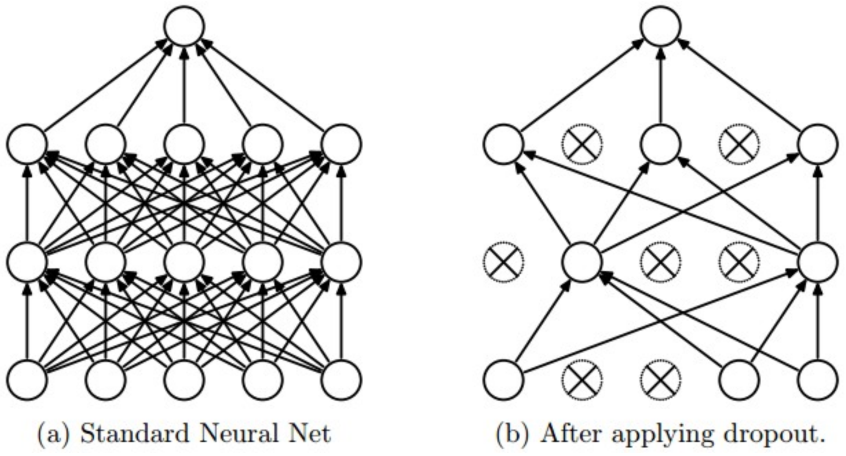
[5] Matias Valdenegro-Toro and Daniel Saromo Mori. A Deeper Look into Aleatoric and Epistemic Uncertainty Disentanglement. arXiv:2204.09308 [cs.LG], April 2022.

[6] M. Havasi, R. Jenatton, S. Fort, J. Z. Liu, J. Snoek, B. Lakshminarayanan, A. M. Dai, and D. Tran. Training Independent Subnetworks for Robust Prediction. In *Proceedings of the International Conference on Learning Representations (ICLR 2021)*, arXiv:2010.06610v2 [cs.LG], Aug 2021.

[7] A. Kendall, V. Badrinarayanan, and R. Cipolla. Bayesian SegNet: Model Uncertainty in Deep Convolutional Encoder-Decoder Architectures for Scene Understanding. arXiv:1511.02680v2 [cs.CV], 10 Oct 2016.

[8] A. Kendall, Y. Gal, and R. Cipolla. Multi-Task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics. arXiv:1705.07115v3 [cs.CV], 24 Apr 2018.

Appendix



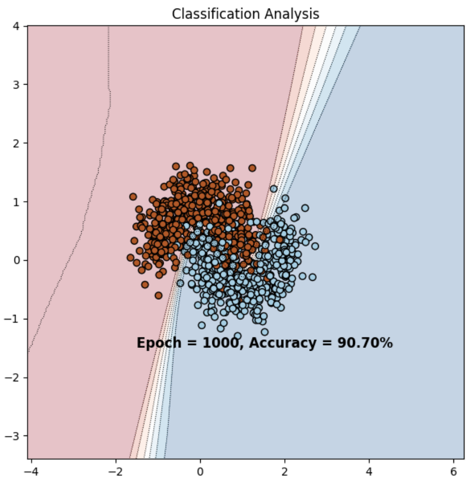


Figure 1: Using Gaussian VI

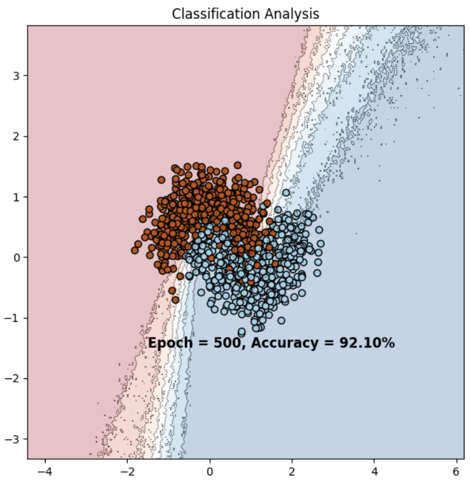


Figure 2: Using MC dropout

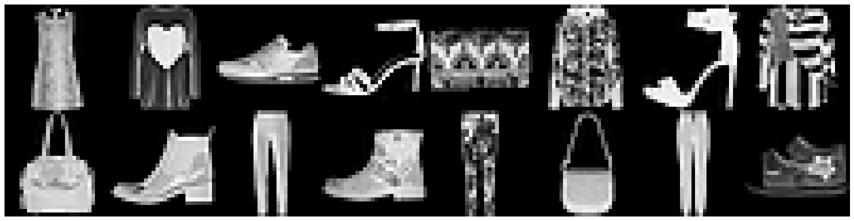


Figure 3: Inputs MNIST Fashion

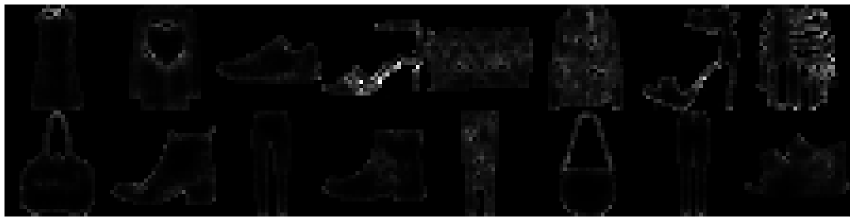


Figure 4: Aleatoric MNIST Fashion

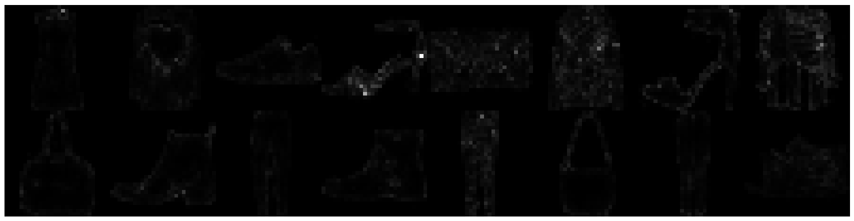


Figure 5: Epistemic MNIST Fashion

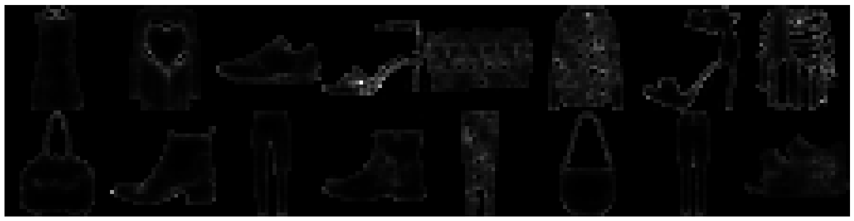


Figure 6: Combined MNIST Fashion

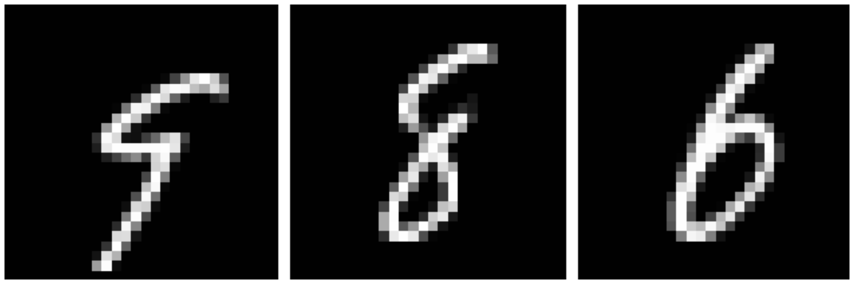


Figure 7: Real digits

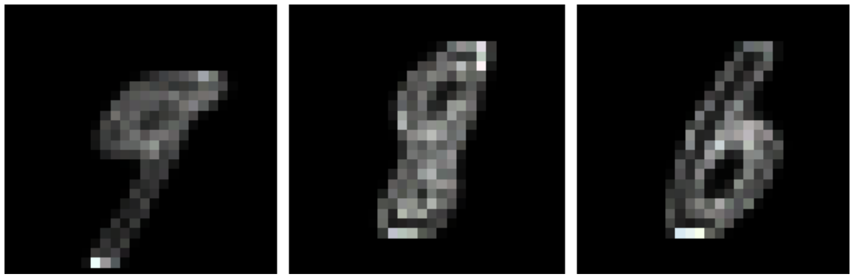


Figure 8: aleatoric uncertainty

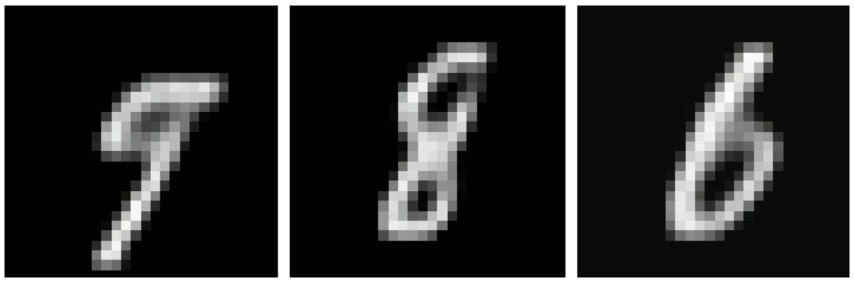


Figure 9: generated with aleatoric uncertainty

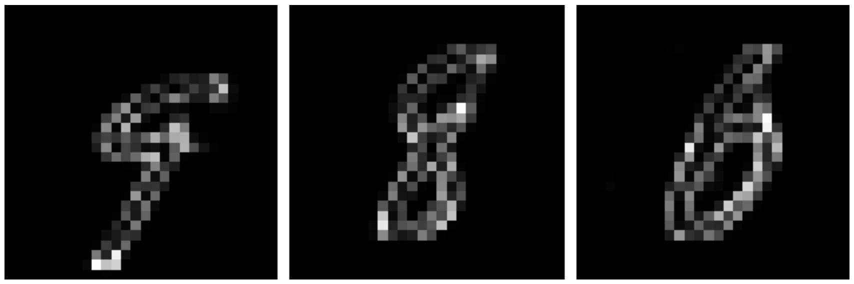


Figure 10: epistemic uncertainty

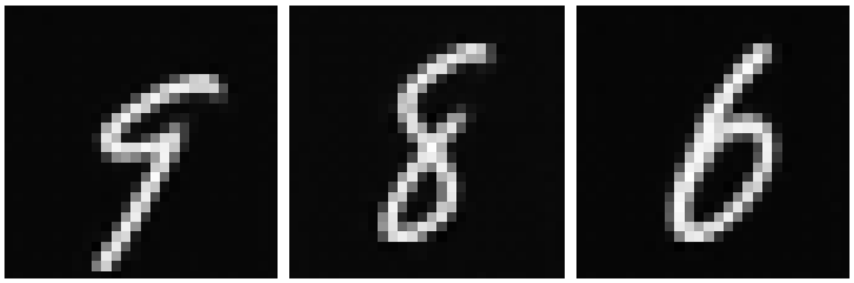


Figure 11: generated with epistemic uncertainty

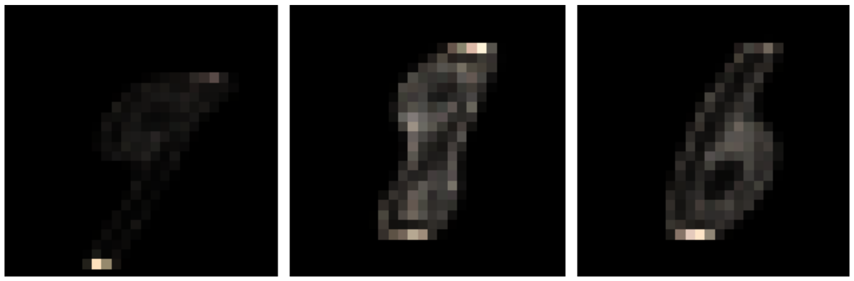


Figure 12: combined aleatoric uncertainty

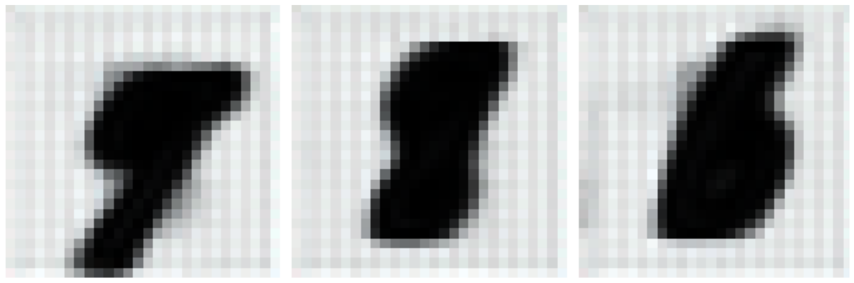


Figure 13: combined epistemic uncertainty

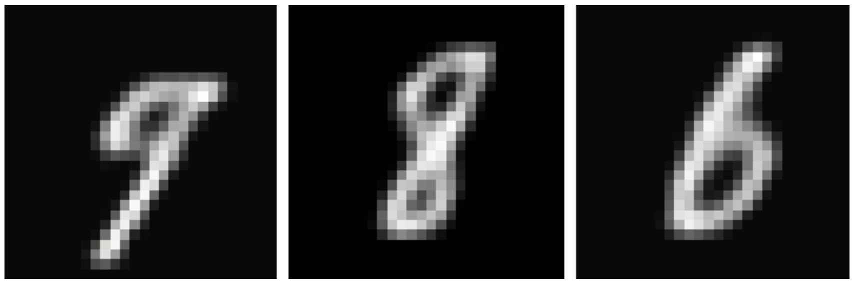


Figure 14: generated with combined uncertainties