Can Monte Carlo dropout help you to sample better examples for Active Learning?

Kirillov Daniil
daniil.kirillov@studenti.unitn.it

CiMEC - Center for Mind and Brain Science
University of Trento

January 30, 2026

Abstract

Monte Carlo dropout is a technique that can turn our neural network model into a makeshift ensemble of models without an additional training effort. It allows us to represent a model's uncertainty in the prediction by processing results of multiple forward passes.

In this project I am trying to use this effect in order to implement entropy-based uncertainty sampling strategy based on dropout-enabled inference and compare it with other simple uncertainty sampling strategies in order to find out: does my implementation choose better training examples for Active Learning?

1 Introduction

In the modern era of Big Data, bigger models, and the biggest AI investment, it is easy to lose sight of approaches which intend to achieve more results with less effort. For some tasks we do not have the huge amount of available data for training, or we do, but the cost of labeling it is quite substantial.

While some approaches (like Data Augmentation) seek to multiply that small datasets we have, Active Learning tries to make compiling training datasets more intentional, by enabling the model we are training to "tell" us, which examples would be the most useful to it. This approach doesn't save us training time (since we need an already trained model in order to find out what data it "needs" next), but can save us the labeling budget.

How does the model "tell" us what data it "needs"? Uncertainty Modeling is used for that purpose. We assume that the more uncertain our model is on its prediction for a specific example, the more useful this example would be in the training dataset. Simple strategies (such as entropy or margin) estimate

Table 1: Network Architecture

Conv2d(1, 32, kernel_size=(3, 3), stride=(1, 1), padding=same)
ELU()
BatchNorm2d(32)
MaxPool2d(kernel_size=(2, 2), stride=2, padding=0)
ConcreteDropout(1e-5, 1e-3)
Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=same)
ELU()
BatchNorm2d(32)
MaxPool2d(kernel_size=(2, 2), stride=2, padding=0)
ConcreteDropout(1e-5, 1e-3)
Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1), padding=same)
ELU()
BatchNorm2d(64)
MaxPool2d(kernel_size=(2, 2), stride=2, padding=0)
Flatten()
ConcreteDropout(1e-5, 1e-3)
Linear(in_features=1024, out_features=128)
ELU()
ConcreteDropout(1e-5, 1e-3)
Linear(in_features=128, out_features=32)
ELU()
Linear(in_features=32, out_features=10)

the model's uncertainty based on the class probabilities the model returns as prediction.

But simple strategies might not be the best performing ones and substantial effort was made in order to improve the uncertainty estimation available. One of the approaches is Monte Carlo Dropout [1], which proposes to use multiple predictions for the same input with dropout layers left on as an estimate for the model's uncertainty for these predictions. Dropout layers randomly switch off neurons with a given probability, allowing us to use multiple forward passes through the same model to emulate an ensemble of models. Then we can use the average probability between those passes as our final prediction. The idea here is that if those quasi-models agree on their predictions, then uncertainty of the combined prediction would be low, and vice versa.

In this work, I implement Monte Carlo dropout and use the highest entropy of the prediction (**EntropyD**) as a sampling strategy for Active Learning. I compare the performance of this sampling strategy with other simple strategies, such as the highest Variation Ratios (**VarRatio**), the lowest **Margin** and the highest **Entropy** (without dropout).

The code for this project with additional details can be found on GitHub [2].

2 The Dataset

I chose to use The Street View House Numbers (SVHN) [3] dataset for this experiment.

SVHN contains cropped 32x32 pixel images obtained from house numbers in Google Street View images centered on the number the model needs to recognise. It is a classical dataset for this type of task, but more interesting (and more hard) than MNIST.

In order to prepare images for training they were turned into greyscale and normalized. I have balanced the size of classes using the size of the smallest class, resulting in 46590 samples in the training set and 15950 samples in the test set. The dataset was shuffled with a fixed random seed of 42.

3 The Model

3.1 Architecture

The architecture I used was inspired by the second architecture from [4], because it seems to demonstrate a decent performance. I have changed the configuration of dense layers, to make them more triangular, changed activation functions to ELU to combat "dying RELU" problem and added dropout layers after every layer with activation (except the penultimate one) in order to implement Monte Carlo Dropout.

According to [5], removing dropout before the ultimate layer can improve model's uncertainty estimation. In my tests it has also improved classification accuracy for a tenth of a percentage point.

Full architecture is presented in Table 1.

3.2 Concrete Dropout

The calibration of uncertainty with Monte Carlo dropout heavily depends on the dropout probabilities of layers. One approach to this problem is to use a hyperparameter tuning in order to pick optimal parameters. This approach is computation heavy, since it requires training multiple neural networks, and also not well fit for the Active Learning task, since optimal dropout probability might be different for different steps of training.

Another approach named Concrete Dropout [6] seeks to solve this problem. It treats dropout probability of a layer as a trainable parameter using the model's backpropagation. This probability changes during training trying to settle on an optimal value.

I have also implemented changes for Concrete Dropout from [7] in order to improve numerical stability of its calculations.

3.3 Training regime and hyperparameters

Following training regime and hyperparameters were used:

- SGD as optimizer following recommendations from [8], with learning rate of 0.001 and momentum of 0.9.
- 20% of the training set (9318 samples) were used as the validation set. The size of batches during training was 64 samples.
- For greater reproducibility the random seed of Pytorch was fixed (with value of 42) before initialization of every model.
- Every model was trained for 200 epochs and the best performing model on the validation set was saved as the best checkpoint.
- For Concrete Dropout weight regularization of 1e-5 and dropout regularization of 1e-3 were chosen as best performing during training on the full dataset.
- For Active Learning, the initial train dataset size of 1000 random samples was chosen and 1000 samples were added during each step (as advised in [9]). 36 steps of Active Learning were performed (not counting the initial one).
- Variation Ratios, Margin and Entropy for sampling were calculated using the ModAL library [10].

4 Evaluation metrics

Knowing only classification performance is not enough to compare sampling strategies between each other. We are interested not only in achieving high performance, but also in achieving it on a smaller labeled samples budget. Keeping this in mind, I have used following evaluation metrics:

1. Accuracy vs budget curves allow us to visually comprehend the Active Learning process and compare models between each other.
2. Highest classification performance, because this is the training objective. Higher is better.
3. Area Under Budget Curve (**AUBC**) - represents how close our model's performance is to the ideal model (which is always right from the start) and how quickly its performance rises. Higher is better.
4. True performance of the selection strategy (**TP**). If the sampling strategy is good, the model's performance would not only rise quickly, but also would not go down during Active Learning steps (because the model only receives samples that are very informative to it). **TP** [11] tries to take this into account by calculating the difference between relative performance gain and relative performance loss during active learning. Higher is better.

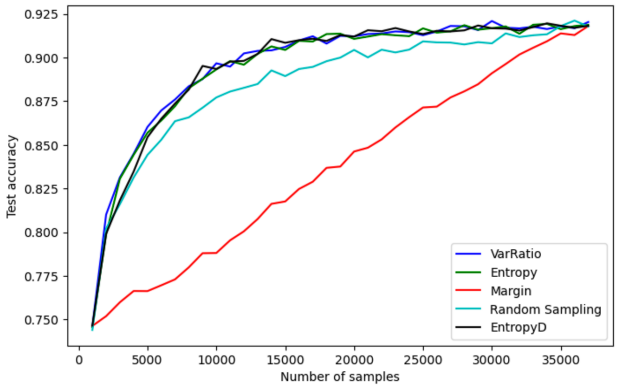

Figure 1: Accuracy vs budget curves.

5 Experimental setup

Following sampling strategies were compared during the experiment:

1. **Random sampling**
2. The highest Variation Ratios (**VarRatio**) - selects data x whose most likely label has lowest posterior probability.
3. The lowest **Margin** - selects data x whose two most likely labels have smallest difference in posterior probabilities.
4. The highest **Entropy** - selects data x that maximize the predictive entropy.
5. The highest Entropy with Monte Carlo dropout (**EntropyD**) - selects data x that maximize the predictive entropy of averaged predictions after T forward passes with dropout. For purpose of this experiment I chose $T = 50$.

6 Results

The accuracy vs budget curves representing the Active Learning process for models with every sampling strategy can be seen in Figure 1.

The first noticeable thing is the performance of the **Margin** sampling strategy. It is considerably worse than the performance of other strategies (even **Random baseline**). The reason for this is an interaction between my code and ModAl library - for all other sampling strategies examples bigger uncertainties are those with higher scores, but for **Margin** are those with lower scores. I did expect the ModAL library to invert scores automatically, but that doesn't happen. As a result, whenever I use **Margin** for sampling I am adding to the train dataset examples not with the highest uncertainty, but with the lowest one.

Table 2: First 10 steps of Active Learning

Sampling strategy	Highest accuracy	AUBC	TP
Random	0.88	0.844	0.115
VarRatio	0.897	0.858	0.116
Margin	0.795	0.77	0.034
Entropy	0.898	0.855	0.13
EntropyD	0.898	0.854	0.117

The performance of **VarRatio**, **Entropy** and **EntropyD** seems comparable, with **Random sampling** being expectedly worse.

Looking at accuracy vs budget curves we can see that best performing models rise up quickly and then start meandering, going up and down in performance. For this reason it doesn't make much sense to analyse all 36 steps. So next I will divide analysis into two parts.

Results for the first 10 steps of active learning can be seen in Table 2. During this part of training performance quickly goes up and models compete with each other on achieving better performance on a smaller budget. We can see that **EntropyD** achieves performance close to **VarRatio** and **Entropy**, but its accuracy over budget grows a bit slower.

Results for the first 20 steps of active learning can be seen in Table 3. During this part of training performance of models sometimes goes down, but overall is getting close to the highest possible one. **EntropyD** achieves highest performance at this part, but **VarRatio** is still better overall in how its performance changes over Active Learning steps.

7 Discussion and Limitations

My ultimate answer to the title question - does not seem like it.

Adding Monte Carlo dropout doesn't seem to have an effect on performance of the highest Entropy sampling strategy, which is in agreement with the results in [12]. In fact, **Entropy** and **EntropyD** demonstrate very close performance, which suggests that multiple forward passes have little effect on the entropy of the prediction.

This project have plenty of limitations, such as: only one dataset was used for testing, only one neural network architecture was tested, Monte Carlo Dropout was tested only with entropy, other uncertainty sampling strategies that work with Monte Carlo dropout weren't tested, uncertainty estimation of Monte Carlo dropout might not have been well calibrated (one interesting solution to that is in [13]). Addressing those limitations would be an interesting next step in continuing the research in this topic.

Table 3: First 20 steps of Active Learning

Sampling strategy	Highest accuracy	AUBC	TP
Random	0.904	0.868	0.127
VarRatio	0.913	0.883	0.129
Margin	0.848	0.798	0.078
Entropy	0.914	0.881	0.12
EntropyD	0.916	0.88	0.121

8 AI usage disclosure

In the purpose of this project an AI model (Gemini 3) was used in order to format references. Without absolute success.

References

[1] Gal, Y., Ghahramani, Z. (2016). *Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning.* In Proceedings of the 33rd International Conference on Machine Learning (pp. 1050-1059).

[2] Kirillov Daniil. *MonteCarloSampling* GitHub Repository. Available at: https://github.com/nealkafree/MonteCarloSampling

[3] Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., Ng, A. Y. (2011). *Reading Digits in Natural Images with Unsupervised Feature Learning.* In NIPS Workshop on Deep Learning and Unsupervised Feature Learning.

[4] Miah, M. R. (2023). *CNN-SVHN Street View Housing Number Digit.* Kaggle Code Notebook. Available at: https://www.kaggle.com/code/mdriponmiah/cnn-svhn-street-view-housing-number-digit

[5] Suhan S., Junhee S. (2025). *Improving Monte Carlo dropout uncertainty estimation with stable output layers.* In Neurocomputing (Vol. 661).

[6] Yarin, G., Jiri, H., Alex, K.(2017). *Concrete Dropout.*

[7] Korsch, D., Shadaydeh, M., Denzler, J. (2025). *Simplified Concrete Dropout - Improving the Generation of Attribution Masks for Fine-grained Classification.* In International Journal of Computer Vision 133 (pp. 5857–5871).

[8] Ji, Y., Kaestner, D., Wirth, O., Wressnegger, C. (2023). *Randomness Is the Root of All Evil: More Reliable Evaluation of Deep Active Learning.* In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (pp. 3943-3952).

[9] Edrina, G., Jiankang, D., Ismail, E. (2025). *Deep active learning: A reality check* In Pattern Recognition Letters (Vol. 197, pp. 188-194).

[10] *MODAL: A Modular Active Learning Framework in Python.* Documentation. Available at: https://modal-python.readthedocs.io/

[11] Oscar, R., Abdulrahman, H. A., Sebastián, V. (2018). *Statistical comparisons of active learning strategies over multiple datasets* In Knowledge-Based Systems (Vol. 145, pp. 274-288).

[12] Zhan, X., Wang, H., Huang, S. J., Chen, J. L., Wang, Y., Du, Y. (2022). *A Comparative Survey of Deep Active Learning*

[13] Asgharnezhad, H., Shamsi, A., Alizadehsani, R., Mohammadi, A., Alinejad-Rokny, H. (2025). *Enhancing Monte Carlo Dropout Performance for Uncertainty Quantification.*