Segmentation and Uncertainty

AD - 4

Exercises

1. What is the loss function for a deterministic segmentation model? What is the connection to standard image classification?

   Answewer: We usually use a cross entropy function as its loss function. The cross-entropy can be expressed as below:

   $L = -\sum_{classes} y_{true} log(y_{pred})$

   In standard image classification, our goal is to classify the whole picture. The cross entropy loss function used is to measure the difference between the image label and model output. In segmentation, our goal is to classify every pixel. And the cross entropy function here depicts the sum of the classification difference of every pixel.

2. What is the difference between aleatoric and epistemic uncertainty in deep learning?

   Answer:

   Aleatoric uncertainty is about the task and dataset itself. There are always noise in the data given and aleatoric uncertainty is to measure this kind of uncertainty. It cannot be reduced by giving more training data.

   Epistemic uncertainty is about the model. It is to measure how reliable the prediction given by the model is. For example, if the model only knows how to classify cats and dogs, then the classification result of a picture of pig must be unreliable. Epistemic uncertainty can be reduced by giving more and more kinds of traning data.

3. To capture epistemic uncertainty, we can model the segmentation network using principles of Bayesian deep learning, where a probability distribution is maintained over each of the individual network weights. Dropout provides a way to approximate such a Bayesian neural network.

   How does sampling in the Dropout model during test time allow for epistemic uncertainty estimation ?

   Answer:

   Bayesian neural networks aim to learn a posterior over weights $P(W|X, Y)$

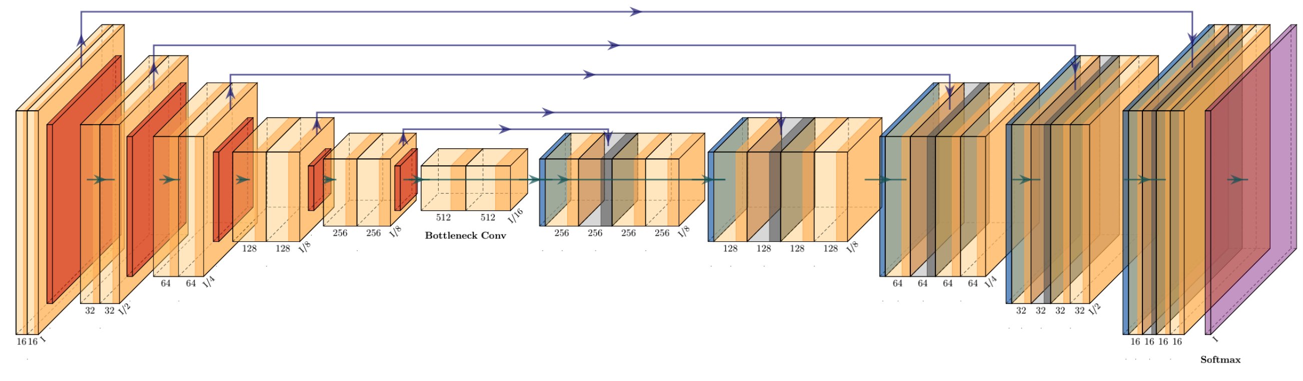
according to bayes therem: $P(W|X,Y) = \frac{P(Y|X,W)P(W)}{P(Y|X)}$

but $P(Y|X)$ is intractable. But we have $P(Y|X) = \sum P(Y|X,W_i)P(W_i)$

Dropout allows sampling from W, and generate a approximation of $P(Y|X)$

Software Lab

1. **Our model:** Unet

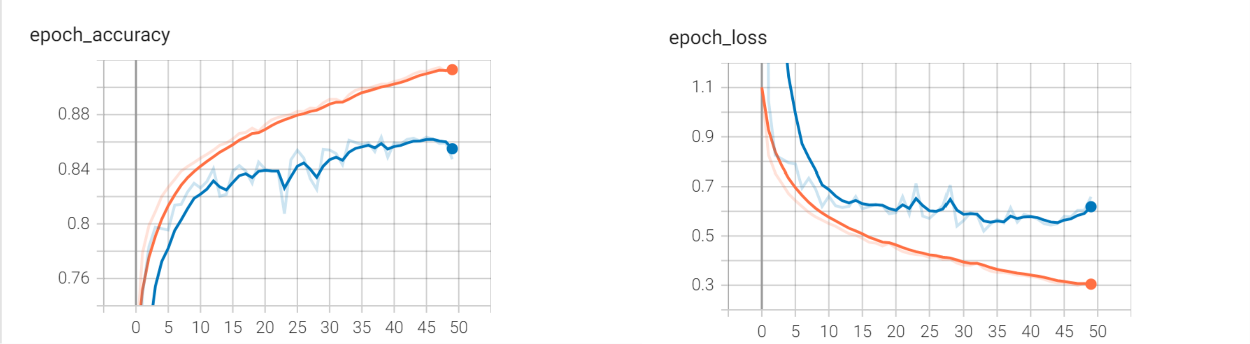


2. **loss function:** sparse categorical cross entropy

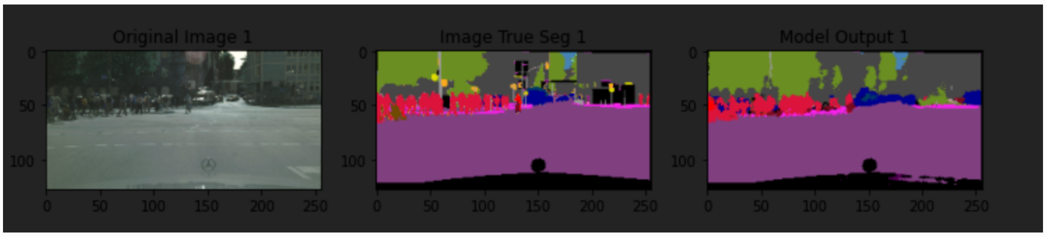
   Because our Y is not one-hot labeled and the model output is one-hot labeled, we use sparse categorical cross entropy.

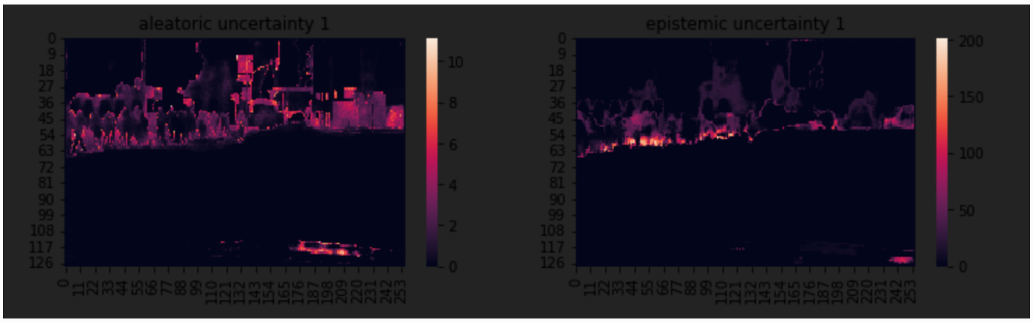
   Optimizer: Adam

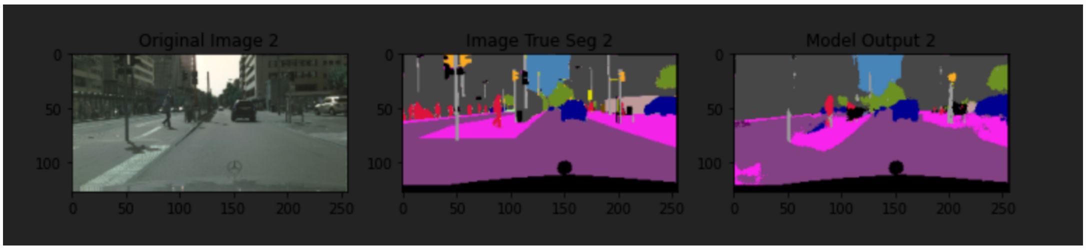
3. Training log

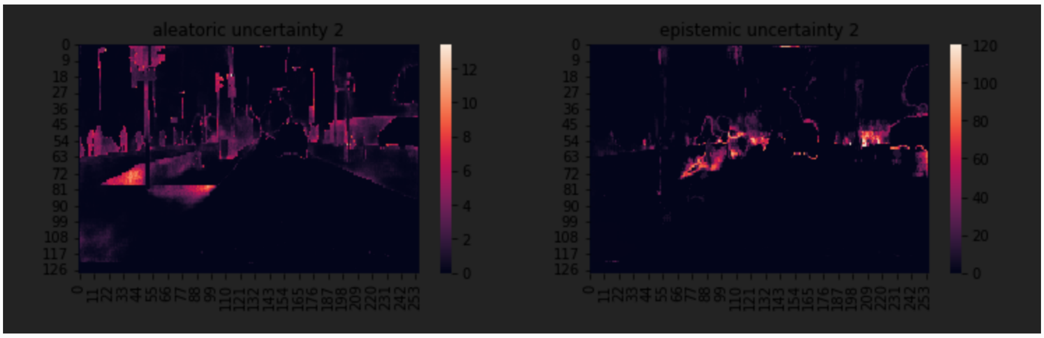


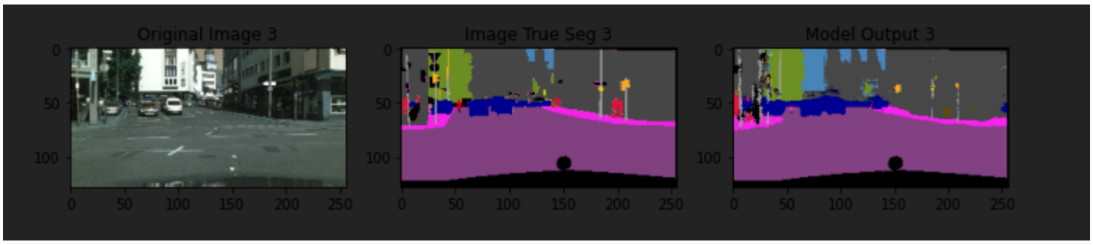
4. Result presentation

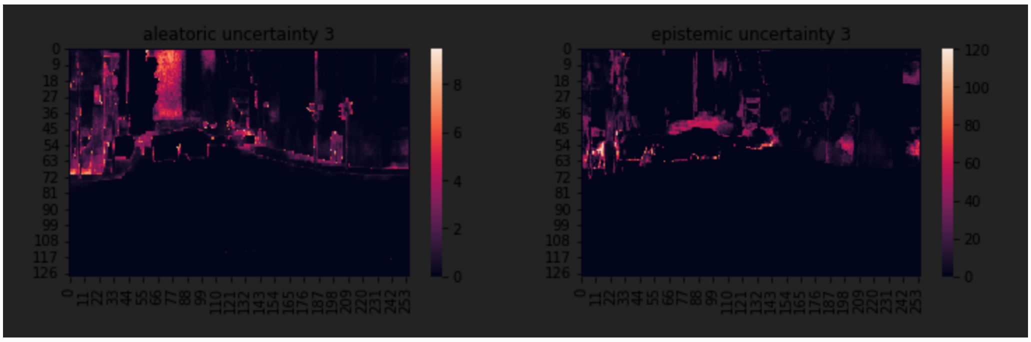


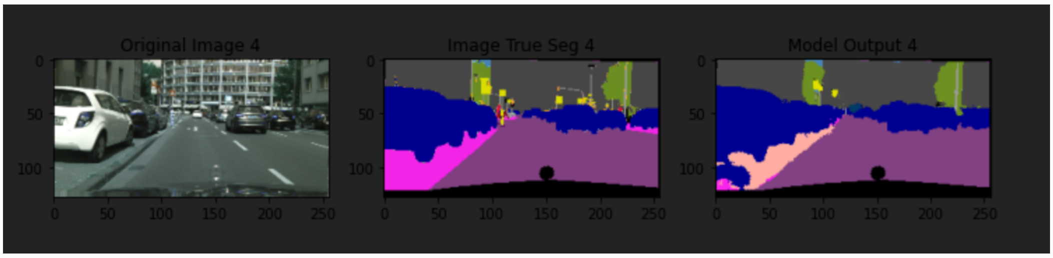

aleatoric uncertainty 1
epistemic uncertainty 1
0 9 18 27 36 45 54 63 72 81 90 99 108 117 126
0 11 22 33 44 55 66 77 88 99 110 121 132 143 154 165 176 187 198 209 220 231 242 253
0 2 4 6 8 10
0 50 100 150 200

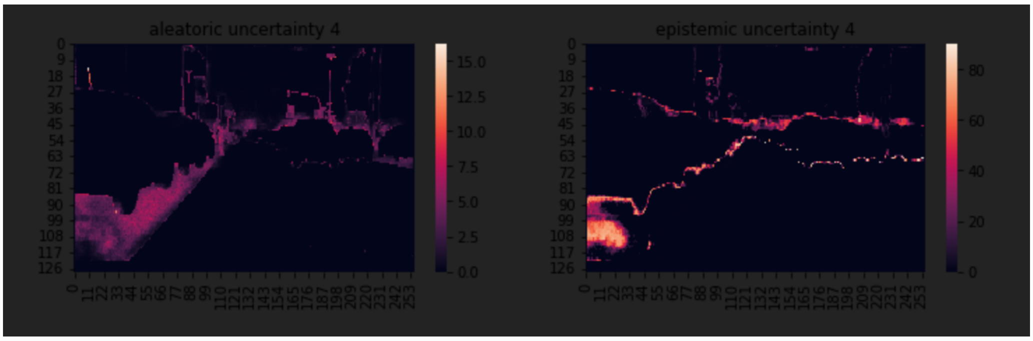

Original Image 2
Image True Seg 2
Model Output 2
0 50 100
0 50 100 150 200 250

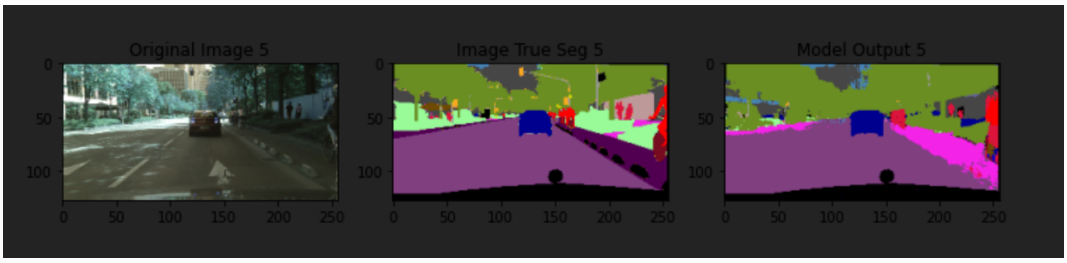

aleatoric uncertainty 2
epistemic uncertainty 2
0 9 18 27 36 45 54 63 72 81 90 99 108 117 126
0 11 22 33 44 55 66 77 88 99 110 121 132 143 154 165 176 187 198 209 220 231 242 253
0 2 4 6 8 10 12
0 20 40 60 80 100 120

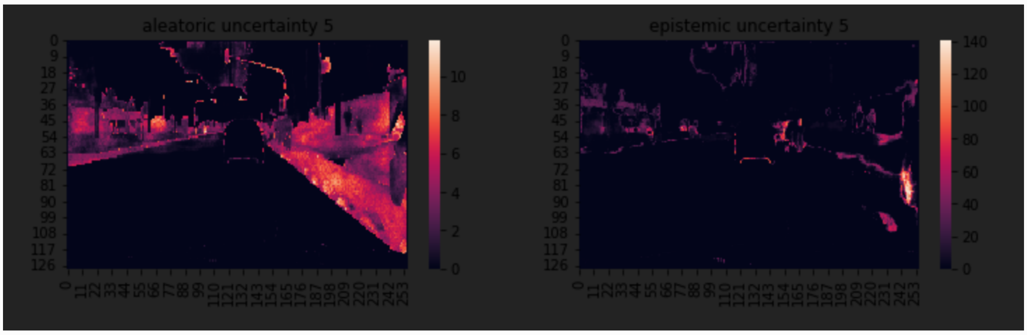

Original Image 3
Image True Seg 3
Model Output 3
0 50 100
0 50 100 150 200 250


aleatoric uncertainty 3
epistemic uncertainty 3
0 9 18 27 36 45 54 63 72 81 90 99 108 117 126
0 11 22 33 44 55 66 77 88 99 110 121 132 143 154 165 176 187 198 209 220 231 242 253
0 2 4 6 8
0 20 40 60 80 100 120


Original Image 4
Image True Seg 4
Model Output 4
0 50 100
0 50 100 150 200 250







5. Result Ananysis

It can be seen that the segmentation result is good as a whole. The model can be improved if increasing the filter numbers and using uncertainties to adjust the output. However, we have a limited hardware condition and such a large model like this is really hard to train.