Rethinking Bayesian Deep Learning Methods for Semi-Supervised Volumetric Medical Image Segmentation — Supplementary Material

Jianfeng Wang
Department of Computer Science
University of Oxford
jianfeng.wang@cs.ox.ac.uk

Thomas Lukasiewicz
Department of Computer Science
University of Oxford
thomas.lukasiewicz@cs.ox.ac.uk,

A. Derivation of ELBO (5)

Proof. As for the joint distribution $P(X, Y)$, we have:

$$\begin{aligned}
logP(X,Y) &= log\int_Z P(X,Y,Z)dZ \\
&= log\int_Z \frac{P(X,Y,Z)}{Q(Z|X)}Q(Z|X)dZ \\
&\geq \mathbb{E}_Q[log\frac{P(X,Y,Z)}{Q(Z|X)}] \\
&= \mathbb{E}_Q[log\frac{P(Y|X,Z)P(X|Z)P(Z)}{Q(Z|X)}] \\
&= \mathbb{E}_Q[logP(Y|X,Z) + logP(X|Z)] - \mathbb{E}_Q[log(\frac{Q(Z|X)}{P(Z)})],
\end{aligned} \tag{1}$$

where $Q(Z|X)$ is the variational distribution, and $\mathbb{E}_Q$ denotes the expectation over $Q(Z|X)$. □

B. Proof of Theorem 1

Proof. Consider two D-dimensional multivariate Gaussian distributions $\mathcal{N}(\mu_1, \Sigma_1)$ and $\mathcal{N}(\mu_2, \Sigma_2)$. Then, the product of their PDFs can be written as follows:

$$\begin{aligned}
&(2\pi)^{-D}det[\Sigma_1]^{-\frac{1}{2}}det[\Sigma_2]^{-\frac{1}{2}}e^{-\frac{1}{2}[(x-\mu_1)^T\Sigma_1^{-1}(x-\mu_1)+(x-\mu_2)^T\Sigma_2^{-1}(x-\mu_2)]} \\
&= C_1e^{-\frac{1}{2}(x^T(\Sigma_1^{-1}+\Sigma_2^{-1})x-x^T(\Sigma_1^{-1}\mu_1+\Sigma_2^{-1}\mu_2)-(\mu_1^T\Sigma_1^{-1}+\mu_2^T\Sigma_2^{-1})x+(\mu_1^T\Sigma_1^{-1}\mu_1+\mu_2^T\Sigma_2^{-1}\mu_2))} \\
&= C_1e^{-\frac{1}{2}(x^T(\Sigma_1^{-1}+\Sigma_2^{-1})x-x^T(\Sigma_1^{-1}+\Sigma_2^{-1})(\Sigma_1^{-1}+\Sigma_2^{-1})^{-1}(\Sigma_1^{-1}\mu_1+\Sigma_2^{-1}\mu_2)-(\mu_1^T\Sigma_1^{-1}+\mu_2^T\Sigma_2^{-1})x+(\mu_1^T\Sigma_1^{-1}\mu_1+\mu_2^T\Sigma_2^{-1}\mu_2))} \\
&= C_1e^{-\frac{1}{2}(x^T(\Sigma_1^{-1}+\Sigma_2^{-1})(x-(\Sigma_1^{-1}+\Sigma_2^{-1})^{-1}(\Sigma_1^{-1}\mu_1+\Sigma_2^{-1}\mu_2))-(\mu_1^T\Sigma_1^{-1}+\mu_2^T\Sigma_2^{-1})x+(\mu_1^T\Sigma_1^{-1}\mu_1+\mu_2^T\Sigma_2^{-1}\mu_2))} \\
&= C_1e^{-\frac{1}{2}(x^T(\Sigma_1^{-1}+\Sigma_2^{-1})(x-(\Sigma_1^{-1}+\Sigma_2^{-1})^{-1}(\Sigma_1^{-1}\mu_1+\Sigma_2^{-1}\mu_2))-(\mu_1^T\Sigma_1^{-1}+\mu_2^T\Sigma_2^{-1})x+(\mu_1^T\Sigma_1^{-1}+\mu_2^T\Sigma_2^{-1})(\Sigma_1^{-1}+\Sigma_2^{-1})^{-1}(\Sigma_1^{-1}\mu_1+\Sigma_2^{-1}\mu_2)+C_2)} \\
&= C_1e^{-\frac{1}{2}(x^T(\Sigma_1^{-1}+\Sigma_2^{-1})(x-(\Sigma_1^{-1}+\Sigma_2^{-1})^{-1}(\Sigma_1^{-1}\mu_1+\Sigma_2^{-1}\mu_2))-(\mu_1^T\Sigma_1^{-1}+\mu_2^T\Sigma_2^{-1})(x-(\Sigma_1^{-1}+\Sigma_2^{-1})^{-1}(\Sigma_1^{-1}\mu_1+\Sigma_2^{-1}\mu_2))+C_2)} \\
&= C_1e^{-\frac{1}{2}(x^T(\Sigma_1^{-1}+\Sigma_2^{-1})(x-(\Sigma_1^{-1}+\Sigma_2^{-1})^{-1}(\Sigma_1^{-1}\mu_1+\Sigma_2^{-1}\mu_2))-(\mu_1^T\Sigma_1^{-1}+\mu_2^T\Sigma_2^{-1})(\Sigma_1^{-1}+\Sigma_2^{-1})^{-1}(\Sigma_1^{-1}+\Sigma_2^{-1})(x-(\Sigma_1^{-1}+\Sigma_2^{-1})^{-1}(\Sigma_1^{-1}\mu_1+\Sigma_2^{-1}\mu_2))+C_2)} \\
&= C_1e^{-\frac{1}{2}((x^T-(\mu_1^T\Sigma_1^{-1}+\mu_2^T\Sigma_2^{-1})(\Sigma_1^{-1}+\Sigma_2^{-1})^{-1})(\Sigma_1^{-1}+\Sigma_2^{-1})(x-(\Sigma_1^{-1}+\Sigma_2^{-1})^{-1}(\Sigma_1^{-1}\mu_1+\Sigma_2^{-1}\mu_2))+C_2)} \\
&= C_1e^{-\frac{1}{2}((x-(\Sigma_1^{-1}+\Sigma_2^{-1})^{-1}(\Sigma_1^{-1}\mu_1+\Sigma_2^{-1}\mu_2))^T(\Sigma_1^{-1}+\Sigma_2^{-1})(x-(\Sigma_1^{-1}+\Sigma_2^{-1})^{-1}(\Sigma_1^{-1}\mu_1+\Sigma_2^{-1}\mu_2))+C_2)} \\
&= C_3e^{-\frac{1}{2}(x-(\Sigma_1^{-1}+\Sigma_2^{-1})^{-1}(\Sigma_1^{-1}\mu_1+\Sigma_2^{-1}\mu_2))^T(\Sigma_1^{-1}+\Sigma_2^{-1})(x-(\Sigma_1^{-1}+\Sigma_2^{-1})^{-1}(\Sigma_1^{-1}\mu_1+\Sigma_2^{-1}\mu_2))},
\end{aligned} \tag{2}$$

where $C_1 = (2\pi)^{-D}det[\Sigma_1]^{-\frac{1}{2}}det[\Sigma_2]^{-\frac{1}{2}}$, C_2 is a constant for aborting the terms used for completing the square relative to x, and $C_3 = C_1e^{-\frac{1}{2}C_2}$. The last formula of Eq. (2) is an unnormalized Gaussian curve with mean $\mu_* = (\Sigma_1^{-1}+\Sigma_2^{-1})^{-1}(\Sigma_1^{-1}\mu_1 + \Sigma_2^{-1}\mu_2)$ and covariance $\Sigma_* = (\Sigma_1^{-1}+\Sigma_2^{-1})^{-1}$. Then, we replace the covariance matrix with the precision matrix, and the mean and the precision of the unnormalized Gaussian curve becomes $\mu_* = \Lambda_*^{-1}(\Lambda_1\mu_1 + \Lambda_2\mu_2)$ and $\Lambda_* = \Lambda_1 + \Lambda_2$, respectively. The theorem can be directly extended to the product of more than two multivariate Gaussian PDFs. □

C. Proof of Corollary 1

Proof. Consider two D-dimensional multivariate Gaussian distributions $P_1 = \mathcal{N}(\mu_1, \Sigma_1)$ and $P_2 = \mathcal{N}(\mu_2, \Sigma_2)$. Then, $\mathbb{E}_{P_1}[log(P_1/P_2)]$ is:

$$\begin{aligned}
&\mathbb{E}_{P_1}[logP_1 - logP_2] \\
&= \frac{1}{2}\mathbb{E}_{P_1}[-logdet[\Sigma_1] - (x-\mu_1)^T\Sigma_1^{-1}(x-\mu_1) + logdet[\Sigma_2] + (x-\mu_2)^T\Sigma_2^{-1}(x-\mu_2)] \\
&= \frac{1}{2}(log\frac{det[\Sigma_2]}{det[\Sigma_1]} + \mathbb{E}_{P_1}[-(x-\mu_1)^T\Sigma_1^{-1}(x-\mu_1) + (x-\mu_2)^T\Sigma_2^{-1}(x-\mu_2)]) \\
&= \frac{1}{2}(log\frac{det[\Sigma_2]}{det[\Sigma_1]} + \mathbb{E}_{P_1}[-tr[\Sigma_1^{-1}\Sigma_1] + tr[\Sigma_2^{-1}(xx^T - 2x\mu_2^T + \mu_2\mu_2^T)]]) \\
&= \frac{1}{2}log\frac{det[\Sigma_2]}{det[\Sigma_1]} - \frac{D}{2} + \frac{1}{2}\mathbb{E}_{P_1}[tr[\Sigma_2^{-1}(xx^T - 2x\mu_2^T + \mu_2\mu_2^T)]] \\
&= \frac{1}{2}log\frac{det[\Sigma_2]}{det[\Sigma_1]} - \frac{D}{2} + \frac{1}{2}\mathbb{E}_{P_1}[tr[\Sigma_2^{-1}((x-\mu_1)(x-\mu_1)^T + 2\mu_1x^T - \mu_1\mu_1^T - 2x\mu_2^T + \mu_2\mu_2^T)]] \\
&= \frac{1}{2}log\frac{det[\Sigma_2]}{det[\Sigma_1]} - \frac{D}{2} + \frac{1}{2}tr[\Sigma_2^{-1}(\Sigma_1 + \mu_1\mu_1^T - 2\mu_2\mu_1^T + \mu_2\mu_2^T)] \\
&= \frac{1}{2}log\frac{det[\Sigma_2]}{det[\Sigma_1]} - \frac{D}{2} + \frac{1}{2}tr[\Sigma_2^{-1}\Sigma_1] + \frac{1}{2}tr[\mu_1^T\Sigma_2^{-1}\mu_1 - 2\mu_1^T\Sigma_2^{-1}\mu_2 + \mu_2^T\Sigma_2^{-1}\mu_2)] \\
&= \frac{1}{2}log\frac{det[\Sigma_2]}{det[\Sigma_1]} - \frac{D}{2} + \frac{1}{2}tr[\Sigma_2^{-1}\Sigma_1] + \frac{1}{2}(\mu_2-\mu_1)^T\Sigma_2^{-1}(\mu_2-\mu_1).
\end{aligned} \tag{3}$$

Based on our assumption in the paper body (left column, lines 442–445), P_1 follows a multivariate Gaussian distribution, and P_2 follows a multivariate standard normal distribution. The above equation becomes $-\frac{1}{2}logdet[\Sigma_1] - \frac{D}{2} + \frac{1}{2}tr[\Sigma_1] + \frac{1}{2}\mu_1^T\mu_1$. Then, we replace μ_1 and Σ_1 with the mean and precision of Theorem 1, obtaining the following equation:

$$\frac{1}{2}(-log\,det[(\sum_{i=0}^{n-1}\Lambda_i)^{-1}] + tr[(\sum_{i=0}^{n-1}\Lambda_i)^{-1}] + (\sum_{i=0}^{n-1}\Lambda_i\mu_i)^T(\sum_{i=0}^{n-1}\Lambda_i)^{-2}(\sum_{i=0}^{n-1}\Lambda_i\mu_i) - D).$$

□

Type	Configuration
3D Conv	#In-C: 3, #F: 64, K: 3 × 3 × 3, S: 1 × 1 × 1, P: 1 × 1 × 1
3D Conv	#In-C: 64, #F: 64, K: 3 × 3 × 3, S: 1 × 1 × 1, P: 1 × 1 × 1
3D Max-Pooling	K: 2 × 2 × 1, S: 2 × 2 × 1, P: 0
3D Conv	#In-C: 64, #F: 128, K: 3 × 3 × 3, S: 1 × 1 × 1, P: 1 × 1 × 1
3D Conv	#In-C: 128, #F: 128, K: 3 × 3 × 3, S: 1 × 1 × 1, P: 1 × 1 × 1
3D Max-Pooling	K: 2 × 2 × 1, S: 2 × 2 × 1, P: 0
3D Conv	#In-C: 128, #F: 256, K: 3 × 3 × 3, S: 1 × 1 × 1, P: 1 × 1 × 1
3D Conv	#In-C: 256, #F: 256, K: 3 × 3 × 3, S: 1 × 1 × 1, P: 1 × 1 × 1
3D Max-Pooling	K: 2 × 2 × 1, S: 2 × 2 × 1, P: 0
3D Conv	#In-C: 256, #F: 512, K: 3 × 3 × 3, S: 1 × 1 × 1, P: 1 × 1 × 1
3D Conv	#In-C: 512, #F: 512, K: 3 × 3 × 3, S: 1 × 1 × 1, P: 1 × 1 × 1
3D Max-Pooling	K: 2 × 2 × 1, S: 2 × 2 × 1, P: 0
3D Conv	#In-C: 512, #F: 512, K: 3 × 3 × 3, S: 1 × 1 × 1, P: 1 × 1 × 1
3D Conv	#In-C: 512, #F: 512, K: 3 × 3 × 3, S: 1 × 1 × 1, P: 1 × 1 × 1

Table 1. Network configuration of the 3D-UNet encoder. Each 3D convolution kernel is followed by an instance normalization layer [3] and a ReLU function, which are omitted for simplicity.

D. Configuration of LRL

LRL contains two 3D-UNet encoders that have the same network configuration (shown in Table 1), and two 3D-UNet decoders (shown in Tables 2 and 3). Note that "#In-C", "#F", "K", "S", and "P" in these tables denote the number of channels of input, the number of filters, the kernel size, the stride, and the padding size, respectively. The setting of "K", "S", and "P" are written in the format "*Height* × *Width* × *Depth*". The scale factor denotes the multiplier to the height, the width, and the depth, when the upsampling operation takes place. To obtain the mean vector and the covariance matrix of each slice, another two small networks were used (shown in Table 4), and they both receive the output of the 3D-UNet encoder. One will generate the mean vector, and the other one will generate a vector, denoted as v, which is used to calculate the covariance matrix with

Type	Configuration
3D Up-sampling	Scale factor: $2 \times 2 \times 1$
3D Conv	#In-C: 1024, #F: 512, K: $3 \times 3 \times 3$, S: $1 \times 1 \times 1$, P: $1 \times 1 \times 1$
3D Conv	#In-C: 512, #F: 512, K: $3 \times 3 \times 3$, S: $1 \times 1 \times 1$, P: $1 \times 1 \times 1$
3D Up-sampling	Scale factor: $2 \times 2 \times 1$
3D Conv	#In-C: 512, #F: 256, K: $3 \times 3 \times 3$, S: $1 \times 1 \times 1$, P: $1 \times 1 \times 1$
3D Conv	#In-C: 256, #F: 256, K: $3 \times 3 \times 3$, S: $1 \times 1 \times 1$, P: $1 \times 1 \times 1$
3D Up-sampling	Scale factor: $2 \times 2 \times 1$
3D Conv	#In-C: 256, #F: 128, K: $3 \times 3 \times 3$, S: $1 \times 1 \times 1$, P: $1 \times 1 \times 1$
3D Conv	#In-C: 128, #F: 128, K: $3 \times 3 \times 3$, S: $1 \times 1 \times 1$, P: $1 \times 1 \times 1$
3D Up-sampling	Scale factor: $2 \times 2 \times 1$
3D Conv	#In-C: 128, #F: 64, K: $3 \times 3 \times 3$, S: $1 \times 1 \times 1$, P: $1 \times 1 \times 1$
3D Conv	#In-C: 64, #F: 64, K: $3 \times 3 \times 3$, S: $1 \times 1 \times 1$, P: $1 \times 1 \times 1$
3D Conv	#In-C: 64, #F: 3, K: $3 \times 3 \times 3$, S: $1 \times 1 \times 1$, P: $1 \times 1 \times 1$

Table 2. Network configuration of the 3D-UNet decoder that is used for reconstruction. Each 3D convolution kernel is followed by an instance normalization layer [3] and a ReLU function except for the last one, which are omitted for simplicity. The last 3D convolution kernel is followed by a TanH function.

Type	Configuration
3D Up-sampling	Scale factor: $2 \times 2 \times 1$
3D Conv	#In-C: 1536, #F: 512, K: $3 \times 3 \times 3$, S: $1 \times 1 \times 1$, P: $1 \times 1 \times 1$
3D Conv	#In-C: 512, #F: 512, K: $3 \times 3 \times 3$, S: $1 \times 1 \times 1$, P: $1 \times 1 \times 1$
3D Up-sampling	Scale factor: $2 \times 2 \times 1$
3D Conv	#In-C: 512, #F: 256, K: $3 \times 3 \times 3$, S: $1 \times 1 \times 1$, P: $1 \times 1 \times 1$
3D Conv	#In-C: 256, #F: 256, K: $3 \times 3 \times 3$, S: $1 \times 1 \times 1$, P: $1 \times 1 \times 1$
3D Up-sampling	Scale factor: $2 \times 2 \times 1$
3D Conv	#In-C: 256, #F: 128, K: $3 \times 3 \times 3$, S: $1 \times 1 \times 1$, P: $1 \times 1 \times 1$
3D Conv	#In-C: 128, #F: 128, K: $3 \times 3 \times 3$, S: $1 \times 1 \times 1$, P: $1 \times 1 \times 1$
3D Up-sampling	Scale factor: $2 \times 2 \times 1$
3D Conv	#In-C: 128, #F: 64, K: $3 \times 3 \times 3$, S: $1 \times 1 \times 1$, P: $1 \times 1 \times 1$
3D Conv	#In-C: 64, #F: 64, K: $3 \times 3 \times 3$, S: $1 \times 1 \times 1$, P: $1 \times 1 \times 1$
3D Conv	#In-C: 64, #F: 2, K: $3 \times 3 \times 3$, S: $1 \times 1 \times 1$, P: $1 \times 1 \times 1$

Table 3. Network configuration of the 3D-UNet decoder that is used for generating label masks. Each 3D convolution kernel is followed by an instance normalization layer [3] and a ReLU function except for the last one, which are omitted for simplicity. The last 3D convolution kernel is followed by a softmax function.

Type	Configuration
3D Conv	#In-C: 512, #F: 512, K: $3 \times 3 \times 1$, S: $2 \times 2 \times 1$, P: $1 \times 1 \times 0$
Flatten	Output shape: [*depth*, $512 \times \frac{h}{2} \times \frac{w}{2}$]
FC	#In-D: $512 \times \frac{h}{2} \times \frac{w}{2}$, #Out-D: 256

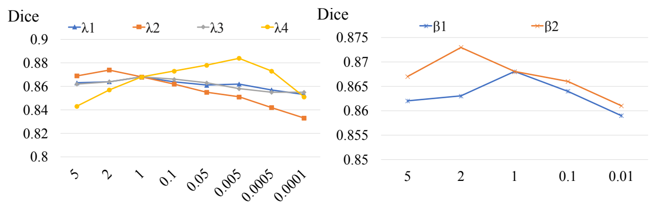
Table 4. Network for generating mean vectors or covariance matrices, which is used after the 3D-UNet encoder. “h” and “w” denote the height and width of the input feature maps, respectively. “#In-D” and “#Out-D” represent the input dimension and the output dimension of the FC layer, respectively.

Type	Configuration
FC	#In-D: 256, #Out-D: $512 \times \frac{h}{2} \times \frac{w}{2}$
Reshape	Output shape: [512, $\frac{h}{2}$, $\frac{w}{2}$, *depth*]
3D Transpose Conv	#In-C: 512, #F: 512, K: $3 \times 3 \times 1$, S: $2 \times 2 \times 1$, P: $1 \times 1 \times 0$

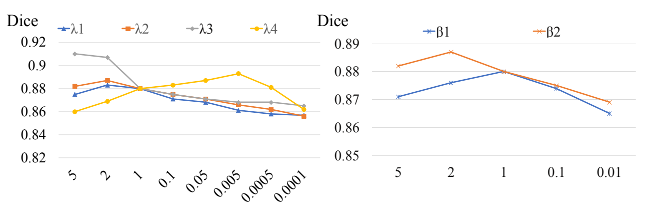
Table 5. Network for transforming the latent representation of each slice into a feature map, which is used before the 3D UNet-decoder. “h” and “w” denote the height and width of the final output feature maps, respectively. “#In-D” and “#Out-D” represent the input dimension and the output dimension of the FC layer, respectively.

$(vv^T)^2 + I_c$, where I_c is a diagonal matrix whose diagonal elements are equal and larger than zero [1]. To transform the latent representation of each slice into feature maps for decoders, we use another small network that is shown in Table 5.

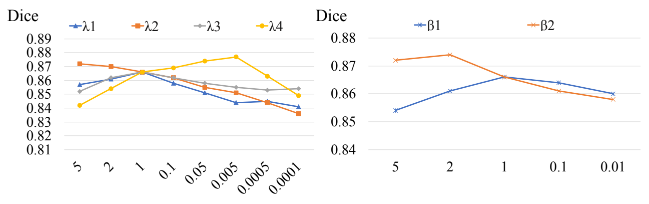
[1]To compute $L_{KL[Q(Z|X)||P(Z))]}$, the log-determinant term requires the input matrix to be positive definite. But the covariance matrix in multivariate Gaussian distribution only needs to be positive semi-definite, which may lead to infinite $L_{KL[Q(Z|X)||P(Z))]}$. Therefore, we add the I_c to solve this issue.



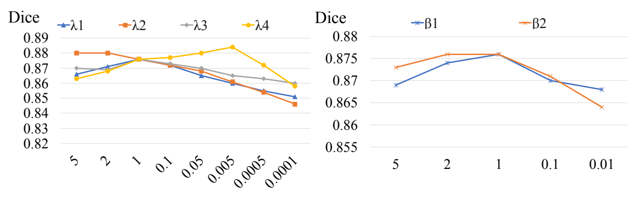
(a) 4 labeled scans and 156 unlabeled scans on the KITS19 dataset.



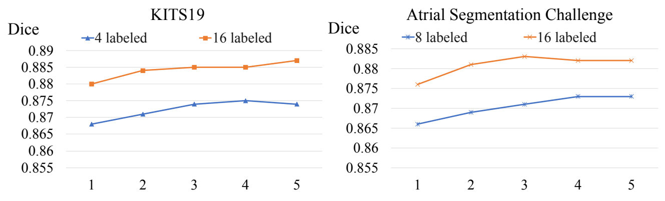
(b) 16 labeled scans and 144 unlabeled scans on the KITS19 dataset.



(c) 8 labeled scans and 72 unlabeled scans on the Atrial Segmentation Challenge dataset.



(d) 16 labeled scans and 64 unlabeled scans on the Atrial Segmentation Challenge dataset.



(e) The relationship between the number of sampled latent representations, namely *M*, and the dice score.

Figure 1. Ablation studies on the coefficient of different loss terms and the number of sampled latent representations.

The workflow of LRL is summarized as follows. For example, the input are volumes with the shape $128 \times 128 \times 32$. Thus, concerning an input volume X, its shape can be written as $3 \times 128 \times 128 \times 32$ (we omit the batch dimension, as we only consider one input volume), where 3 is the number of channels of each slice in the volume, and 32 is the number of slices in the volume, i.e., depth. X is fed to a 3D-UNet encoder, resulting in an output volume with the shape $512 \times 8 \times 8 \times 32$. Then, the output volume is fed to the small networks (Table 4) that are designed for generating mean vectors and the covariance matrices. After the mean vectors and the covariance matrices are obtained, they are further used to produce latent representations (denoted as Z) via the reparameterization trick. Z is further processed by the small network (Table 5) that is designed for transforming the latent representation of each slice into feature maps, resulting in a volume with the shape $512 \times 8 \times 8 \times 32$ (denoted as Z_{map}), and thereafter, Z_{map} is used for reconstructing the input volume X via the 3D-UNet decoder shown in Table 2. At the same time, the input volume is fed to another 3D-UNet encoder to obtain the feature representation with the shape $512 \times 8 \times 8 \times 32$, and then the feature representation is concatenated with Z_{map} along the channel dimension, which is the input to the 3D-UNet decoder shown in Table 3 for generating label masks.

E. Implementation Details

The proposed method was implemented in PyTorch [2]. The configuration of LRL is introduced above, and as for the regular 3D-UNet with MC dropout, we keep the number of parameters similar to VNet in previous works [1, 5, 6] for fair comparison [2], and the MC dropout is inserted into the decoder of the regular 3D-UNet. The GBDL was trained for 120 epochs on the KiTS19 and the Liver Segmentation dataset, and for 240 epochs on the Atrial Segmentation Challenge dataset. In the first half epochs, only the LRL was trained. Then, the LRL was fixed, and the regular 3D-UNet with MC dropout was optimized in the second half epochs. During training, each slice of CT scans was first resized to 256×256, and a 160×160 center region was cropped from the slice. Then, $128 \times 128 \times 32$ volumes were randomly cropped as inputs, where 32 is the depth of the input volumes, i.e., the number of slices. We used the common data augmentation methods, including pixel

[2]Our regular 3D-UNet with MC dropout has 9.60M parameters while VNet has 9.44M parameters. The detailed configuration can be found in our released code.

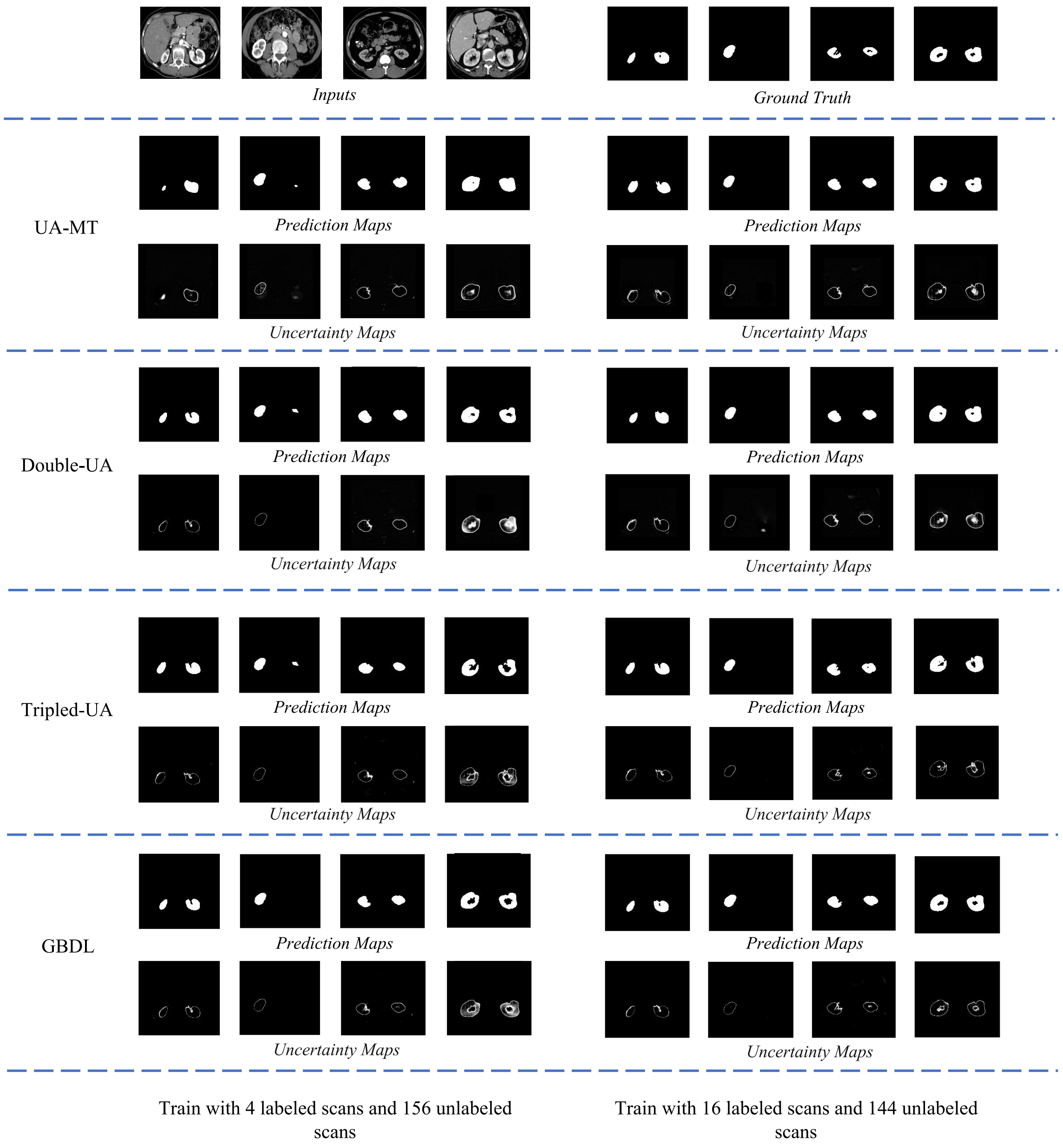


Figure 2. The visualization of some predicted slices from the KiTS19 dataset for different Bayesian deep learning based methods [4–6].

jittering, random rotation, and random horizontal flipping. The ADAM optimizer was used with an initial learning rate of 0.0001 and a batch size of 4, and the cosine learning rate decay strategy was used.

As for the evaluation, we used the sliding window strategy with a stride of $8 \times 8 \times 8$, and the last saved model was used for testing. We took twenty feed-forward passes for each case to get the final result and the corresponding voxel-wise epistemic uncertainty. Each of our experimental results in the paper body was obtained based on five independent runs.

F. Hyperparameter Search

We searched the hyperparameters, including coefficients of different loss terms and the number of sampled latent representations, on the KiTS19 and the Atrial Segmentation Challenge dataset. We further took 10 CTs and 5 CTs from the their test sets, respectively, as their validation set, in order to avoid the situation that all of the test data are involved into the search

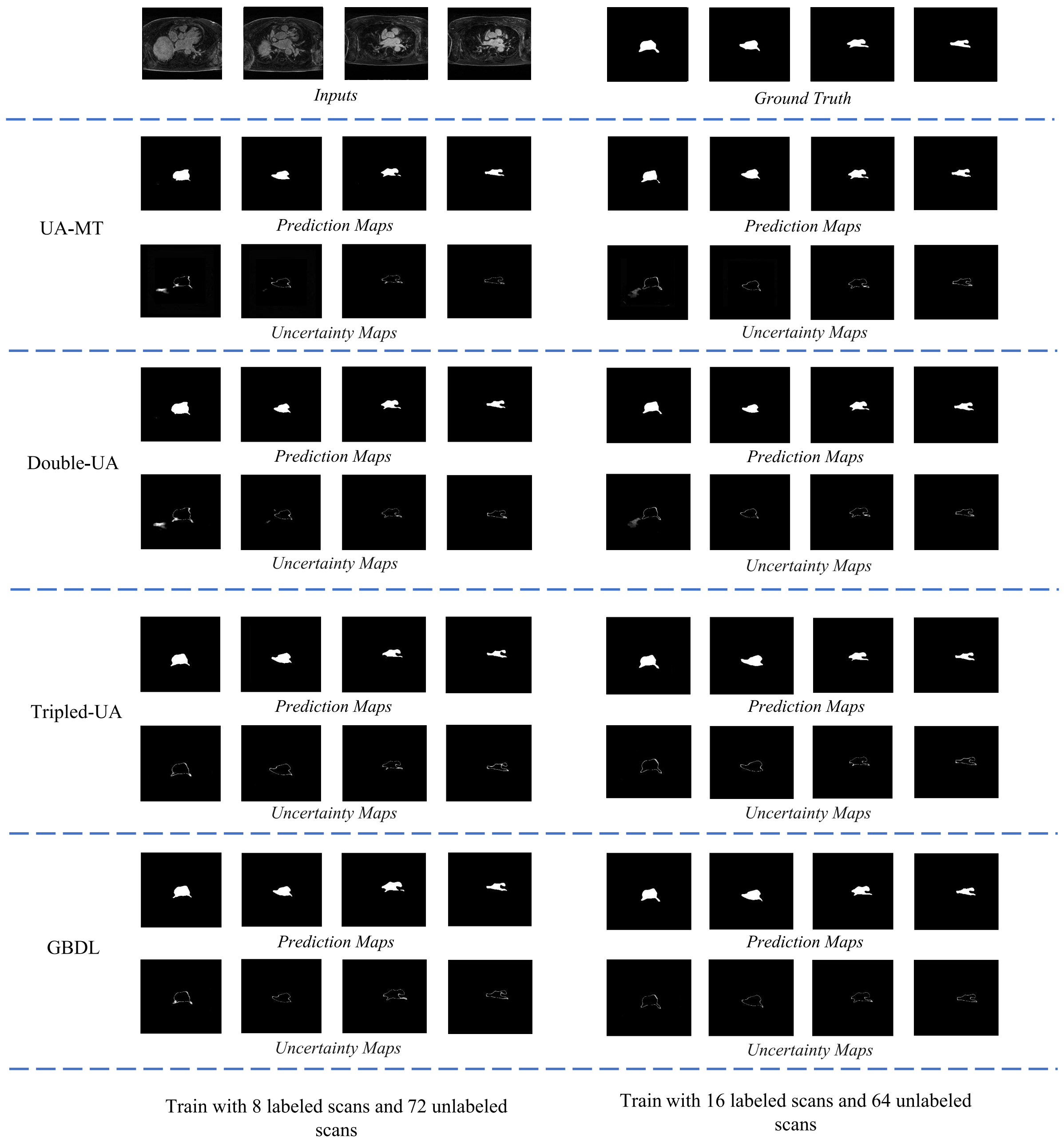


Figure 3. The visualization of some predicted slices from the Atrial Segmentation Challenge dataset for different Bayesian deep learning based methods [4–6].

process [3]. Since it is hard to find the best combination of the coefficients, we resort to using the variable-controlling approach. Initially, all the coefficients are set to 1.0 and only one latent representation is sampled for each feed-forward pass, and then we sequentially changed them one by one. Every time we changed one variable, others were fixed to the initial setting. According to the results in Figure 1, we can conclude that when λ_1, λ_2, and λ_3 are decreased, the performance declines to some extent. In contrast, the performance is not greatly impacted when λ_1, λ_2, and λ_3 are increased, and the performance reaches the peak when λ_1, λ_2, and λ_3 are set to 1.0, 2.0, 1.0, respectively. Concerning λ_4, the performance can reach the peak when it is set to 0.005. As for the coefficient of the terms in L_{Seg}, setting β_1 and β_2 to 1.0 and 2.0, respectively, can achieve the best

[3]Note that previous works do not hold a validation set for tuning hyper-parameters. Therefore, in this work, we took a very small number of data from the test set for the hyper-parameter searching.

result. As for the number of sampled latent representations, we sampled M latent representations to get M pseudo-label maps for an unlabeled sample, and the final pseudo-label map is obtained by taking the average over the M maps. We notice that increasing M can improve the performance, but the performance gain becomes moderate when M is larger than 3. Then, we fixed these coefficients and set M to 5 for all experiments in the paper body.

G. Visualization Results

We visualize some predicted results from the KiTS19 dataset and the Atrial Segmentation Challenge dataset in Figure 2 and Figure 3 for different Bayesian deep learning based methods, including UA-MT [6], Double-UA [5], Tripled-UA [4], and GBDL. Each result contains a prediction map and a corresponding uncertainty map. The uncertainty maps for the other three methods come from their teacher models [4–6], and the brightness of each pixel in an uncertainty map is negatively associated with the model's confidence for predicting that pixel. According to the visualization results, when only a limited number of training data are annotated, GBDL can still find and segment the foreground. Although there are some misclassified regions, uncertainty maps are able to give high uncertainties to them, so that clinicians can further improve the results in a real-world scenario.

References

[1] Xiangde Luo, Jieneng Chen, Tao Song, and Guotai Wang. Semi-supervised medical image segmentation through dual-task consistency. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 8801–8809, 2021.

[2] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. PyTorch: an imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8026–8037, 2019.

[3] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6924–6932, 2017.

[4] Kaiping Wang, Bo Zhan, Chen Zu, Xi Wu, Jiliu Zhou, Luping Zhou, and Yan Wang. Tripled-uncertainty guided mean teacher model for semi-supervised medical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 450–460. Springer, 2021.

[5] Yixin Wang, Yao Zhang, Jiang Tian, Cheng Zhong, Zhongchao Shi, Yang Zhang, and Zhiqiang He. Double-uncertainty weighted method for semi-supervised learning. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 542–551. Springer, 2020.

[6] Lequan Yu, Shujun Wang, Xiaomeng Li, Chi-Wing Fu, and Pheng-Ann Heng. Uncertainty-aware self-ensembling model for semi-supervised 3D left atrium segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 605–613. Springer, 2019.