Chapter 3

Deformable medical image registration

In this chapter, we discuss about different approaches used to attempt deformable medical image registration. We begin with the classical approaches which are then followed by recent deep learning based model. We then explain our proposed Bayesian deep learning framework and finally, we compare our results with recent state-of-the art approach.

3.1 Classical approaches

Deformable image registration is one of the challenging inverse problems in medical imaging. It aims to estimate the underlying DVF considering the distortions across all the pixel locations in the moving image [20]. Here DVF is a vector field containing the displacement vectors for each point in the moving image w.r.t reference image coordinates. The DVF along with appropriate interpolation technique gives us the resampled registered image.

Broad philosophy used by classical approaches as discussed in section 1.1 as well as fig 1.4 is mentioned below,

1.1 Initialize displacement values with zeros i.e $\mathbf{T} = 0$

1.2 Let $\mathrm{I}_{ref}(x, y)$ and $\mathrm{I}_{mov}(x, y)$ be the intensity of reference image and the moving image, respectively.

1.3 Interpolate the moving image with displacement values $\mathbf{T}$.

1.4 The interpolated image (registered image) is then compared with reference image with similarity metric.

1.5 Demons force **U** is calculated such that the similarity is maximized in each iteration.

1.6 $\nabla \mathrm{I}_{ref}(x, y)$ be the gradient of the reference image.

1.7 The demons force **U** is calculated as :

$$\mathbf{U} = \mathbf{U}_1 + \mathbf{U}_2 \tag{3.1}$$

where $\mathbf{U}_1$ is

$$\mathbf{U}_1 = \frac{I_{mov}(x, y) - I_{ref}(x, y)}{|\nabla I_{ref}(x, y)|^2 + \alpha^2 (I_{mov}(x, y) - I_{ref}(x, y))^2} \nabla I_{ref}(x, y) \tag{3.2}$$

and $\mathbf{U}_2$

$$\mathbf{U}_2 = \frac{I_{mov}(x, y) - I_{ref}(x, y)}{|\nabla I_{ref}(x, y)|^2 + \alpha^2 (I_{mov}(x, y) - I_{ref}(x, y))^2} \nabla I_{mov}(x, y) \tag{3.3}$$

1.8 Update displacement value matrix **T** = **T** - **U**

1.9 Interpolate moving image with **T** by interpolaton technique

The equation (3.1) is calculated iteratively during the registration process for a given number of iteration. In active demons algorithm [42], the gradient information of the moving image as a force is introduced into the demons driving force calculation formula. In order to adjust the force strength adaptively in each iteration, a normalization factor α is adopted. Complete algorithm for registering image is mentioned in Algorithm 1.

A smaller value of α can speed up the convergence, however will reduce the accuracy. A larger value of α can achieve higher precision, at slow down the convergence speed. Here the transformation parameters are estimated using the displacement vector field. The DVF is then interpolated with moving image to generate the registered image. Finally, this registered image is compared with fixed image using different similarity metrics. This process is repeated for number of iterations and once the iterations are completed the resulting interpolated image is considered as the registered image. The results obtained by using classical approach is discussed in section 3.4

Algorithm 1 A classical approach for deformable medical image registration [43].

Input: A pair of reference $\mathbf{I_{ref}}$ and moving $\mathbf{I_{mov}}$) image.
Output: Registered moving image $\mathbf{I_{reg}}$ with respect to $\mathbf{I_{ref}}$
Initialization:
α =2.5,
Transformed x coordinates: $T_x = 0$,
Transformed y coordinates: $T_y = 0$,
Gradient of reference image in x direction : $\mathbf{S}_x = \nabla I_{ref_x}(x, y)$,
Gradient of reference image in y direction : $\mathbf{S}_y = \nabla \mathrm{I}_{ref_y}(x, y)$,

```
function REGISTER($\mathbf{I_{ref}}, \mathbf{I_{mov}}$)
    for number of iterations do
        /* Calculate intensity difference between moving and reference image */
        $\mathbf{I}_{diff} = \mathbf{I_{mov}} - \mathbf{I_{ref}}$
        /* Calculate gradients of moving image in x and y direction */
        $\mathbf{M}_x = \nabla I_{mov_x}(x, y)$
        $\mathbf{M}_y = \nabla I_{mov_y}(x, y)$
        /* Calculate demons force in both x and y direction as $U_x$ and $U_y$ * /
        $\mathbf{U}_x = -\mathbf{I}_{diff} * (\frac{S_x}{(S_x^2+S_y^2)+\alpha^2 * I_{diff}^2} + \frac{M_x}{(M_x^2+M_y^2)+\alpha^2 * I_{diff}^2})$
        $\mathbf{U}_y = -\mathbf{I}_{diff} * (\frac{S_y}{(S_x^2+S_y^2)+\alpha^2 * I_{diff}^2} + \frac{M_y}{(M_x^2+M_y^2)+\alpha^2 * I_{diff}^2})$
        /* Calculate transformed displacement values to be applied to x coordinates of moving image as $\mathbf{T}_x$ and $\mathbf{T}_y$ * /
        $\mathbf{T}_x = \mathbf{T}_x + \mathbf{U}_x$
        $\mathbf{T}_y = \mathbf{T}_y + \mathbf{U}_y$
        /* Estimate the registered image $\mathbf{I}_{reg}$ using interpolation technique */
        $\mathbf{I}_{reg}$= interpolate($\mathbf{I}_{mov}, \mathbf{T}_x, \mathbf{T}_y$)
    end for
```

3.2 Deep learning for image registration

In this subsection, we highlight recent development under the deep learning based approaches. We discuss the working of basic deep network for deformable image registration task followed by training/test algorithms. The experimental results are shown in section 4.

3.2.1 Deep convolutional neural network

The deep network used for image registration consists of convolutional neural network (CNN's) which consists of convolutional layers. The CNN takes in an input image patch and assigns weights to important features through backpropagation training. Fig. 3.1 shows the basic convolutional neural network used for classification task, consisting of 4 convolution layer followed by 2 fully connected layer. Here an image patch of size 3×3 is convolved with a filter of size 2×2 to get a resulting image patch of size 2×2. This convolution operation along with pooling layer (typically max pooling), and non linearity function as activation function together constitute one convolution layer.

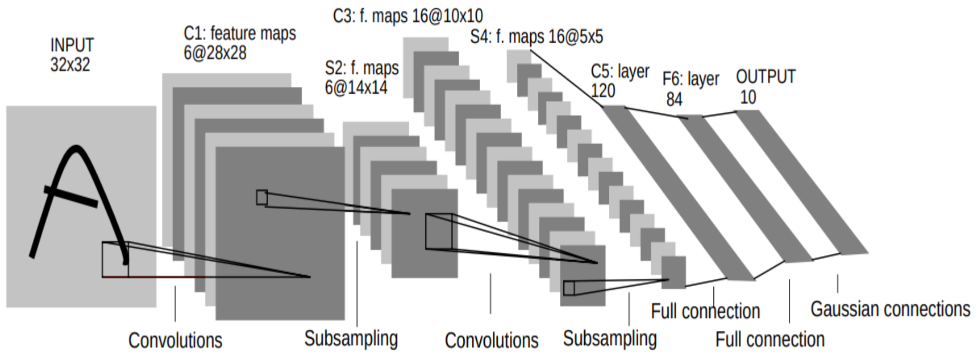

Figure 3.1: An exemplary convolution neural network for classification [26].

In registration task, after a series of convolution operation applied on the input reference and moving image pair, the network outputs the DVF that contains the displacement values to be applied to every pixel coordinates of the moving image . The DVF (transformed coordinates from the CNN) along with moving image is then resampled using interpolation technique e.g., bilinear or bicubic interpolation, in order to get the registered image. Note that, the registration task using deep learning consists of three parts, CNN network, DVF estimation and resampling. This entire network takes in a pair of fixed and moving image pair dataset and outputs the corresponding registered images.

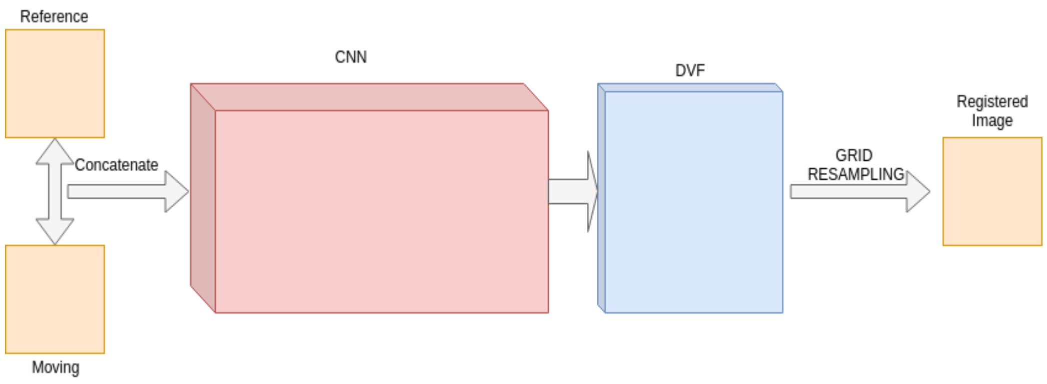

Figure 3.2: A general flowchart showing the working of deep learning algorithm for deformable image registration.

3.2.2 Training and test of the deep network

Deep neural network requires large amount of data to train the network. The given data is first split into training and testing dataset in the ratio of 7:3. This data is then preprocessed before feeding it to the network. The preprocessing steps mainly consists of resizing and applying manual distortions. In this case, we get training and test dataset consisting of image pairs where the reference image is original image (without distortion) and the distorted image is the moving image. The training dataset is then feed to the network. The output of the network, i.e, registered image is compared with the reference image according to given similarity metric such as normalized cross correlation (NCC) and structural similarity index (SSIM). The loss is calculated and then backpropagated [39] to the network using chain rule allowing weights to be updated using weight update rule. This process is repeated for the given number of training iterations allowing the network to learn the weights which appropriately registers the moving image. Once the training is done, the network then relies on its fixed set of weights for the testing phase.

The input image pair of test dataset is then passed through the trained network. Since the network is already trained, the network then registers the moving image. As shown in Fig. 3.2, fixed and moving image pair is passed through the CNN and the CNN outputs DVF using backpropagation training. This DVF along with moving image is then resampled to generate the warped image. There are two components to the process: (i) coordinate transformation (DVF), and (iii) resampling. These two components are collectively known as warping. This warped image is compared to reference image with similarity metric. The overall objective is to maximize the similarity between reference and moving image. The detailed algorithm for training and testing of images is given in Algorithm 2.

Algorithm 2 A basic deep learning algorithm for deformable medical image registration.[3, 50]

Input: A pair of reference $\mathbf{I_{ref}}$ and moving $\mathbf{I_{mov}}$) image.
Output: Registered moving image $\mathbf{I_{reg}}$ with respect to $\mathbf{I_{ref}}$

```
function TRAIN($\mathbf{I_{ref}}, \mathbf{I_{mov}}$)
    $\mathbf{I_{in}}$ = concatenate($\mathbf{I_{ref}}, \mathbf{I_{mov}}$)
    for number of training iterations do
        for number of train data examples do
            /*Apply convolution operation to $\mathbf{I_{in}}$ where $\mathbf{w}$ and $\mathbf{b}$ are weight and bias respectively */
            $\boldsymbol{\psi} = \mathbf{I_{in}} * \mathbf{w} + \mathbf{b}$
            / * $\circ$ is an operator to warp $\mathbf{I_{mov}}$ by $\boldsymbol{\psi}$ to yield $\mathbf{I_{reg}}$ * /
            $\mathbf{I_{reg}} = \mathbf{I_{mov}} \circ \boldsymbol{\psi}$
            /*Calculate the loss*/
            $\mathrm{L}(\mathbf{I_{ref}}, \mathbf{I_{reg}}) = \mathrm{SSIM}(\mathbf{I_{ref}}, \mathbf{I_{reg}}) + \mathrm{MSE}(\mathbf{I_{ref}}, \mathbf{I_{reg}})$
            /* Calculate the vector gradient with respect to $\mathbf{w}$ and $\mathbf{b}$ */
            $\Delta_{\mathrm{w}} = \frac{\delta \mathrm{L}(\mathbf{I_{ref}}, \mathbf{I_{reg}})}{\delta \mathrm{w}}$
            $\Delta_{\mathrm{b}} = \frac{\delta \mathrm{L}(\mathbf{I_{ref}}, \mathbf{I_{reg}})}{\delta \mathrm{b}}$
            /* Update the parameters $\boldsymbol{w}$ and $\boldsymbol{b}$ with learning rate $\alpha$ * /
            $\boldsymbol{w} \leftarrow \boldsymbol{w} - \alpha \Delta_{\boldsymbol{w}}$
            $\boldsymbol{b} \leftarrow \boldsymbol{b} - \alpha \Delta_{\boldsymbol{b}}$

        end for
    end for
end function
```

```
function TEST($\mathbf{I_{ref}}, \mathbf{I_{mov}}$)
    $\mathbf{I_{in}}$ = concatenate($\mathbf{I_{ref}}, \mathbf{I_{mov}}$)
    for number of test data examples do
        /*Apply convolution operation to $\mathbf{I_{in}}$ */
        $\boldsymbol{\psi} = \mathbf{I_{in}} * \mathbf{w} + \mathbf{b}$
        $\mathbf{I_{reg}} = \mathbf{I_{mov}} \circ \boldsymbol{\psi}$
    end for
end function
```

3.3 Proposed Bayesian deep learning

In this subsection, we discuss our proposed Bayesian deep network for deformable medical image registration. Note that, in section 3.2 the conventional CNN [28] relies on fixed set of weights which are updated during training phase and these trained fixed weights are used for testing. The network has limited capacity to capture uncertainity in the data due to fixed weights and also there is high chance of underfitting/ overfitting during training. To overcome this issue, we propose Bayesian deep learning approach where we learn probability distribution over the weights by backpropagation. Given a pair of reference and moving image our objective is to register the moving image onto the reference image coordinates while maintaining the pixel distribution (values) of the moving image.

3.3.1 Proposed Bayesian deep learning network

In the supervised learning set-up, given a set of training examples $\mathbf{D} = (\mathbf{x}_i, \mathbf{y}_i)_i$ where $\mathbf{x}_i$ and $\mathbf{y}_i$ are the i^{th} input and output respectively, the goal is to learn weights $\mathbf{w}$ that shall predict the output $\mathbf{y_{test}}$ corresponding to test input $\mathbf{x_{test}}$. The conventional CNNs can be considered as probabilistic model $P(\mathbf{y}|\mathbf{x},\mathbf{w})$ where $\mathbf{x} \in \mathbb{R}^n$ is input and the CNN predicts the output $\mathbf{y} \in \mathbb{R}^n$ by estimating the set of parameters or weights $\mathbf{w} \in \mathbb{R}^n$. These weights are generally learned by maximum likelihood estimation (MLE) [37], [24], i.e, $\mathbf{w}^{MLE}$ are given by,

$$\begin{aligned} \mathbf{w}^{MLE} &= \arg\max_{\mathbf{w}} \log P(\mathbf{D}|\mathbf{w}), \\ &= \arg\max_{\mathbf{w}} \sum_{i}^{n} \log P(\mathbf{y}_i|\mathbf{x}_i,\, \mathbf{w}), \end{aligned} \tag{3.4}$$

where n is the total number of training examples. The training is typically done by the backpropagation, where one assume that $\log P(\mathbf{D}|\mathbf{w})$ is differentiable in $\mathbf{w}$. Note that, at the end of CNN training, the estimated weights $\mathbf{w}$ are fixed real numbers. Hence, the network has to rely on these hard (fixed) weights (numbers) to predict the output which may lead to severe error especially in medical data.

Unlike the conventional CNNs, Bayesian neural networks estimate the posterior distribution of the weights given the training data $P(\mathbf{w}|D)$ by back propagation [6, 27]. The predictive distribution of output $P(\mathbf{y_{test}})$ of a test data input item $\mathbf{x_{test}}$ is then given by

$$P(\mathbf{y_{test}}|\mathbf{x_{test}}) = \mathbb{E}_{P(\mathbf{w}|\mathbf{D})}[P(\mathbf{y_{test}}|\mathbf{x_{test}}, \mathbf{w})], \tag{3.5}$$

where $\mathbb{E}_{P(\mathbf{w}|\mathbf{D})}$ is conditional expectation over posterior distribution of weights given the training data $\mathbf{D}$.

Each possible configuration of the weights, weighted according to the posterior distribution, makes a prediction about the unknown label given the test data item $\mathbf{x_{test}}$. Thus taking the conditional expectation under the posterior distribution on weights (equation (3.5)) is equivalent to using a large number of neural networks for learning the weights. Fig 3.4 and Fig. 3.4 shows an example of weights learned in classical deep learning and Bayesian approach.

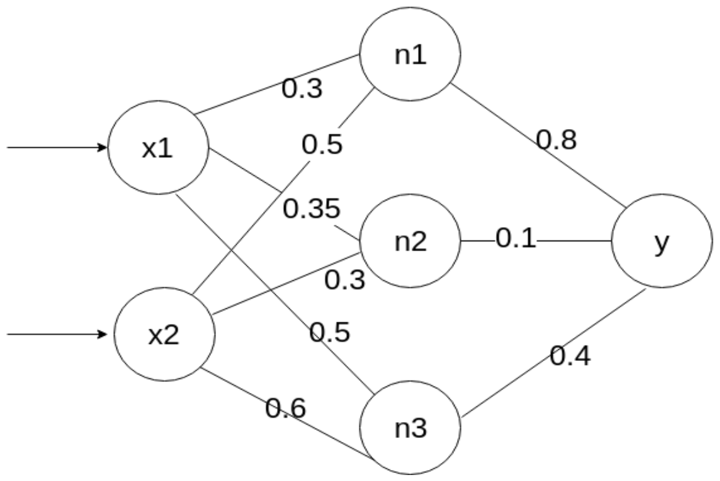

Figure 3.3: An example of deep network where each weight has fixed value as provided by classical backpropagation [50].

In order to overcome the limitations of classical CNNs such as model uncertainity in the medical data, underfitting/ overfitting due to few data

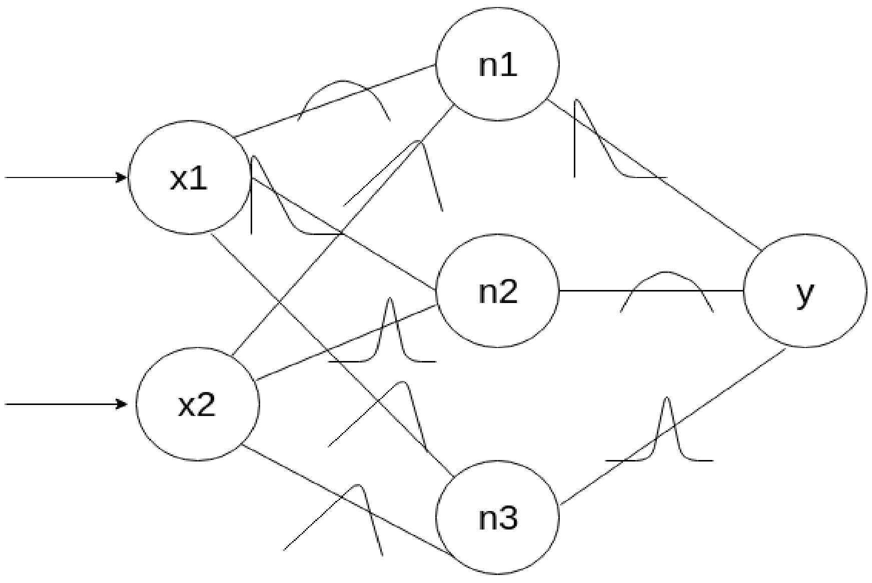

Figure 3.4: An example where each weight has distribution as provided by Bayesian backpropagation [6].

examples and to provide better generalization, we propose the deep Bayesian architecture Fig. 3.5 for achieving deformable medical image registration.

Our proposed network in shown in Fig. 3.5 where a pair of reference and moving image of size 150×150 pixels concatenated into 2-channel 2D image $(150 \times 150 \times 2)$ is then passed through downsampling and upsampling layers. It consists of 5 downsampling and 4 upsampling layers. Each downsampling layer consists of two convolutional layers with filters of size 3×3 where stride is 1 and pooling layer with a stride of 2. Whereas each upsampling layer consists of bilinear interpolation with scale size of 2 followed by two convolutional layers with filters of size 3×3 where stride is 1. In order to incorporate non-linearity, we have used rectified linear units (RELU) [13] and *tanh* as activation functions. As shown in Fig. 3.5, the output of the architecture is the DVF for the moving images. It estimates the geometric displacement required by each pixel coordinate of moving image in order to get it registered with respect to the reference image. The estimated DVF along with grid resampling and bilinear interpolation of moving image gives us the final registered image as shown in Fig. 3.5.

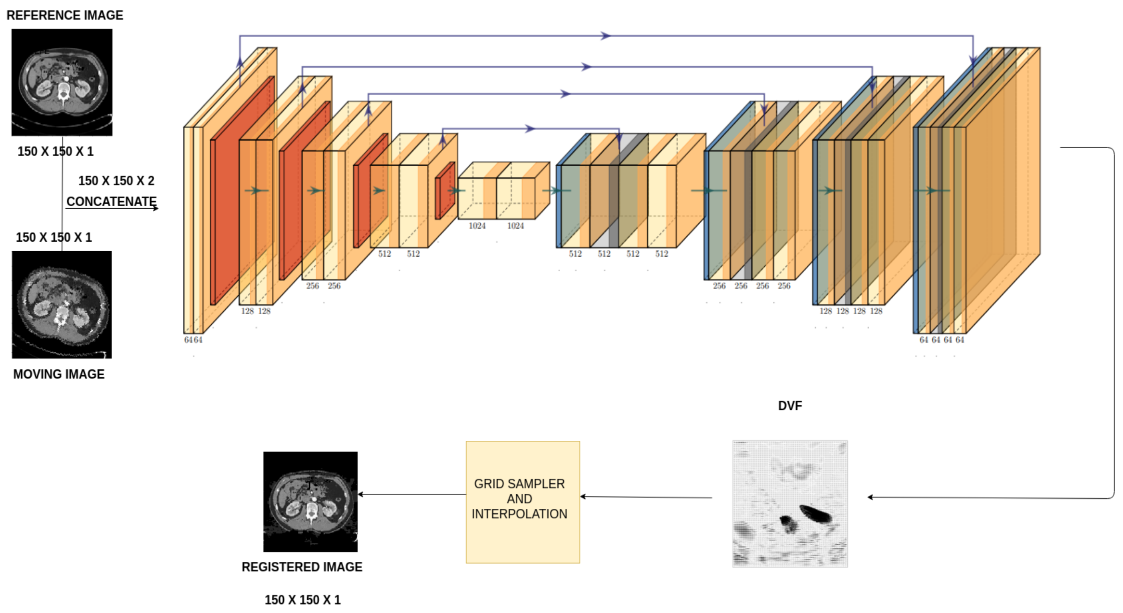

Figure 3.5: Proposed deep Bayesian architecture for deformable medical image registration.

3.3.2 Training/Test of proposed approach

The supervised training of proposed approach as shown in Fig. 3.5 is done as follows: We use a differentiable loss function L that is a combination of structural similarity index (SSIM) [7] and mean squared error (MSE)[51],

$$\mathrm{L}(\mathbf{I_{ref}}, \mathbf{I_{reg}}) = \mathrm{SSIM}(\mathbf{I_{ref}}, \mathbf{I_{reg}}) + \mathrm{MSE}(\mathbf{I_{ref}}, \mathbf{I_{reg}}), \tag{3.6}$$

where $\mathbf{I_{ref}}$ is reference and $\mathbf{I_{reg}}$ is registered image, i.e., ($\mathbf{I_{mov}} \circ \boldsymbol{\psi}$), and

$$\mathrm{MSE}(\mathbf{I_{ref}}, \mathbf{I_{reg}}) = \frac{1}{mn}\sum_{x=1}^{m}\sum_{y=1}^{n}[\mathbf{I_{ref}}(x,y) - \mathbf{I_{reg}}(x,y)]^2, \tag{3.7}$$

$$\mathrm{SSIM}(\mathbf{I_{ref}}, \mathbf{I_{reg}}) = \frac{(2\mu_{I_{ref}}\mu_{I_{reg}} + C_1)(2\sigma_{I_{ref}I_{reg}} + C_2)}{(\mu^2_{I_{ref}} + \mu^2_{I_{reg}} + C_1)(\sigma^2_{I_{ref}} + \sigma^2_{I_{reg}} + C_2)}, \tag{3.8}$$

where $\mathbf{I_{ref}}(x, y)$ and $\mathbf{I_{reg}}(x, y)$ are reference and registered image coordinates, $\mu_{I_{ref}}$ and $\mu_{I_{reg}}$ are means and $\sigma^2_{I_{ref}}$ and $\sigma^2_{I_{reg}}$ are variances of reference and registered image, respectively. Here C_1 and C_2 are constants, m, n are total number of pixels for images.

We have employed batch normalization after every convolution layer, learning rate of 0.0001 and batch size of 1 throughout our training process.

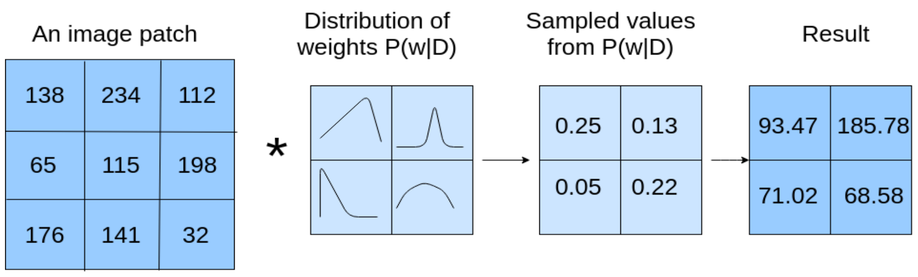

Figure 3.6: An example of internal working of a convolution layer as proposed in Fig. 1. An image patch of size 3×3 is getting convolved with filter/ kernel of size 2×2 where the filter values are sampled from the posterior distribution of weights unlike conventional CNNs.

Since the batch size is 1 the type of gradient descent we employed is stochastic gradient descent. The filter weights are initialized with prior Gaussian distribution and therefore the posterior distribution over the weights are then learned through backpropagation. In order to gain better insight, we now show an example of a convolution layer as proposed in Fig. 3.6.

Once the training of the data is completed, the posterior distribution over the weights are learned. We then sample weights from this distribution to get the filter values as shown in Fig.2 . Finally, this filter is then convolved with image patches as part of convolutional layer. The detailed algorithm is described in Algorithm 3.

3.4 Experiments and Results

In this section we discuss the results obtained for applying the proposed Bayesian deep learning approach for the deformable medical image registration and compared with recent state-of-the-art approaches. In our experiment, we have used CT and MRI medical images . The basic structure of medical image is shown in Fig. 3.7, here the image is structured as volumetric cube consisting of slices and voxels where slices are the cross sectional view of the organs. There are three different views in medical imaging viz. saggital, coronal and axial views.Fig. 3.9 and Fig. 3.10 shows different views of the brain and lungs organs. We have used publicly available lungs Computerized Tomography scan data from the deformable image registration laboratory [8] as well as cardiac Magnetic Resonance Imaging data [2]. MRI's provide more detailed pictures of organ and internal body structures then CT scan [12], [45]. Fig. 3.8 shows the detailed compariosn of both the images. We implement the algorithm in Python and pytorch 1.0 on a machine with Intel

Algorithm 3 Proposed Bayesian deep learning for deformable medical image registration.

Input: A pair of reference $\mathbf{I_{ref}}$ and moving $\mathbf{I_{mov}}$) image.
Output: Registered moving image $\mathbf{I_{reg}}$ with respect to $\mathbf{I_{ref}}$
Assumption 1: Gaussian prior distribution over weights $\mathbf{w}$ i.e $\mathbf{w} = [\mathrm{w}_1, \mathrm{w}_2, ..., \mathrm{w}_n]$ and bias $\mathbf{b} = [\mathrm{b}_1, \mathrm{b}_2, ..., \mathrm{b}_m]$ where $\mathrm{w}_i \sim \mathcal{N}(\mu_i, \sigma_i)$ and $\mathrm{b}_i \sim \mathcal{N}(\mu_i^{'}, \sigma_i^{'})$. Also $\boldsymbol{\mu} = [\mu_1, \mu_2, ..., \mu_n]$ and $\boldsymbol{\mu}^{'} = [\mu_1^{'}, \mu_2^{'}, ..., \mu_m^{'}]$
Assumption 2: Standard deviation $\boldsymbol{\sigma} = [\sigma_1, \sigma_2, ..., \sigma_n]$ and $\boldsymbol{\sigma}^{'} = [\sigma_1^{'}, \sigma_2^{'}, ..., \sigma_n^{'}]$ where $\sigma_i = \log(1 + e^{\rho_i})$ and $\sigma_i^{'} = \log(1 + e^{\rho_i^{'}})$ to ensure non-negative $\sigma_i, \sigma_i^{'}$, and $\rho_i, \rho_i^{'} \in \mathbb{R}$

```
function TRAIN($\mathbf{I_{ref}}, \mathbf{I_{mov}}$)
    $\mathbf{I_{in}}$ = concatenate($\mathbf{I_{ref}}, \mathbf{I_{mov}}$)
    for number of training iterations do
        for number of train data examples do
            sample $\boldsymbol{\epsilon} = [\epsilon_1, \epsilon_2, ..., \epsilon_n]$ and $\boldsymbol{\epsilon}^{'} = [\epsilon_1^{'}, \epsilon'_2, ..., \epsilon_m^{'}]$ where $\epsilon_i, \epsilon_i^{'} \in \mathcal{N}(0, 1)$
            /* Calculate weights ($\mathbf{w}$) and bias ($\mathbf{b}$) where $\odot$ is pointwise multiplication */
            $\mathbf{w} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}$
            $\mathbf{b} = \boldsymbol{\mu}^{'} + \boldsymbol{\sigma}^{'} \odot \boldsymbol{\epsilon}^{'}$
            /*Apply convolution operation to $\mathbf{I_{in}}$ */
            $\boldsymbol{\psi} = \mathbf{I_{in}} * \mathbf{w} + \mathbf{b}$
            / * $\circ$ is an operator to warp $\mathbf{I_{mov}}$ by $\boldsymbol{\psi}$ to yield $\mathbf{I_{reg}}$ * /
            $\mathbf{I_{reg}} = \mathbf{I_{mov}} \circ \boldsymbol{\psi}$
            /*Calculate the loss*/
            $\mathrm{L}(\mathbf{I_{ref}}, \mathbf{I_{reg}}) = \mathrm{SSIM}(\mathbf{I_{ref}}, \mathbf{I_{reg}}) + \mathrm{MSE}(\mathbf{I_{ref}}, \mathbf{I_{reg}})$
            /* Calculate the vector gradient with respect to mean $\boldsymbol{\mu}$ and $\boldsymbol{\rho}$ * /
            $\Delta_{\boldsymbol{\mu}} = \frac{\delta \mathrm{L}(\mathbf{I_{ref}}, \mathbf{I_{reg}})}{\delta \mu}$; $\Delta_{\boldsymbol{\mu}^{'}} = \frac{\delta \mathrm{L}(\mathbf{I_{ref}}, \mathbf{I_{reg}})}{\delta \mu^{'}}$
            $\Delta_{\boldsymbol{\rho}} = \frac{\delta \mathrm{L}(\mathbf{I_{ref}}, \mathbf{I_{reg}})}{\delta \rho}$; $\Delta_{\boldsymbol{\rho}^{'}} = \frac{\delta \mathrm{L}(\mathbf{I_{ref}}, \mathbf{I_{reg}})}{\delta \rho^{'}}$
            /* Update the variational parameters $\boldsymbol{\mu}$ and $\boldsymbol{\rho}$ with learning rate $\alpha$ * /
            $\boldsymbol{\mu} \leftarrow \boldsymbol{\mu} - \alpha \Delta_{\boldsymbol{\mu}}$; $\boldsymbol{\mu}^{'} \leftarrow \boldsymbol{\mu}^{'} - \alpha \Delta_{\boldsymbol{\mu}^{'}}$
            $\boldsymbol{\rho} \leftarrow \boldsymbol{\rho} - \alpha \Delta_{\boldsymbol{\rho}}$; $\boldsymbol{\rho}^{'} \leftarrow \boldsymbol{\rho}^{'} - \alpha \Delta_{\boldsymbol{\rho}^{'}}$

        end for
    end for
end function
```

```
function TEST($\mathbf{I_{ref}}, \mathbf{I_{mov}}$)
    $\mathbf{I_{in}}$ = concatenate($\mathbf{I_{ref}}, \mathbf{I_{mov}}$)
    for number of test data examples do
        sample $\boldsymbol{\epsilon} = [\epsilon_1, \epsilon_2, ..., \epsilon_n]$ and $\boldsymbol{\epsilon}' = [\epsilon_1', \epsilon'_2, ..., \epsilon_m']$ where $\epsilon_i, \epsilon_i' \in \mathcal{N}(0, 1)$
        /* Calculate weights (w) and bias (b) where $\odot$ is pointwise multiplication */
        $\mathbf{w} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}$
        $\mathbf{b} = \boldsymbol{\mu}' + \boldsymbol{\sigma}' \odot \boldsymbol{\epsilon}'$
        /*Apply convolution operation to $\mathbf{I_{in}}$ */
        $\boldsymbol{\psi} = \mathbf{I_{in}} * \mathbf{w} + \mathbf{b}$
        $\mathbf{I_{reg}} = \mathbf{I_{mov}} \circ \boldsymbol{\psi}$
    end for
end function
```

core i5, Nvidia Geforce GTX 1050 4GB with 24 GB RAM .

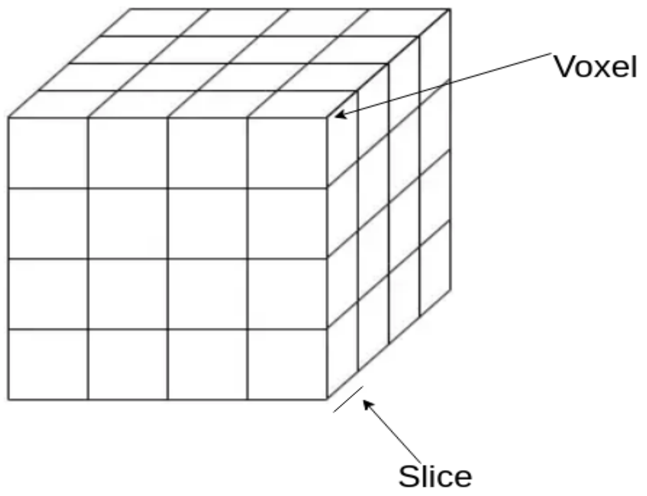

Figure 3.7: The medical image CT/MRI is considered as a volumetric cube consisting of slices and voxel.

3.4.1 Quantitative results on Lung CT dataset

In this subsection we show results of classical , conventional and our approach on the lung CT images [8] and compare it with state-of-the-art approaches.

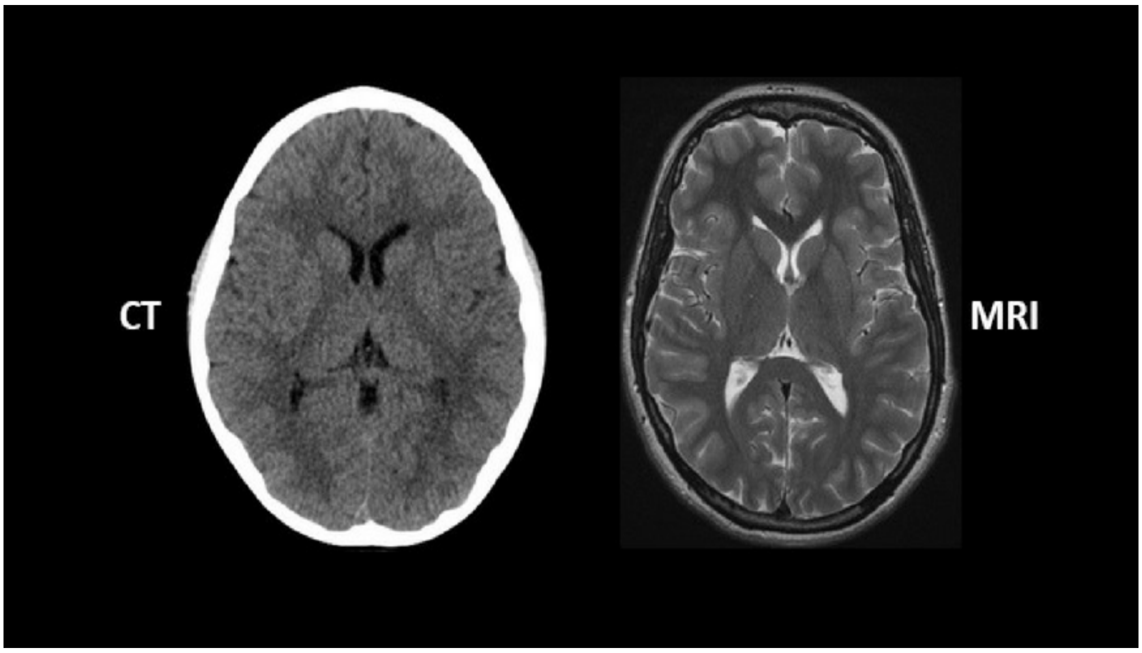

Figure 3.8: CT and MRI image of axial view of brain [46].

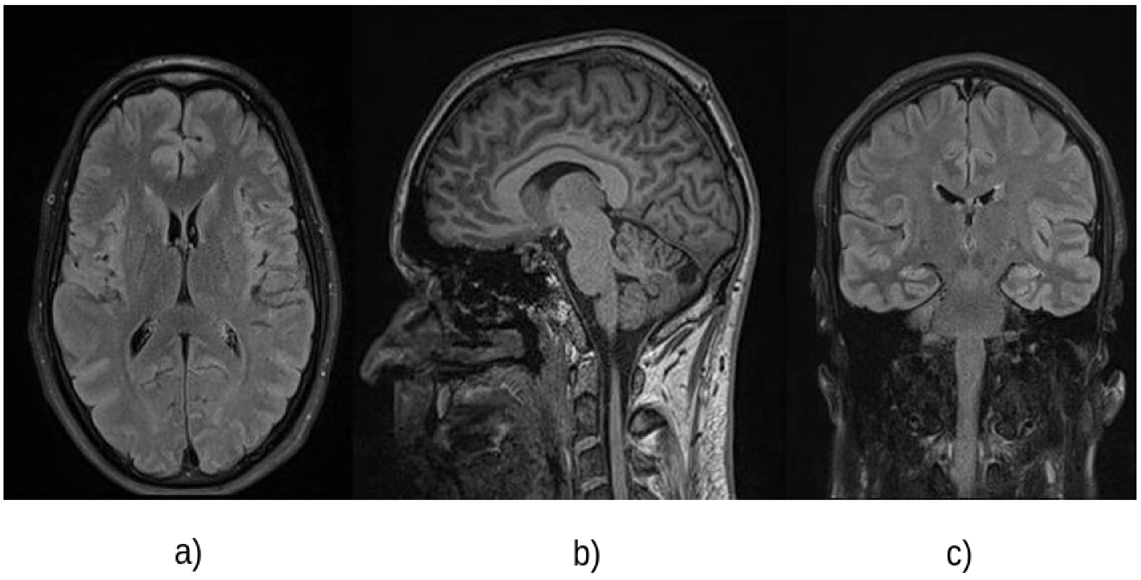

Figure 3.9: Figure showing different views of brain MRI imaging [11](a) axial, (b) sagittal and (c) coronal.

The dataset consists of Thoracic 4DCT images as well as inspiratory and exipratory breath-hold CT image pairs. Entire dataset is arranged in 10 different cases. The images are of size $512 \times 512 \times 128$ voxels with a dimension

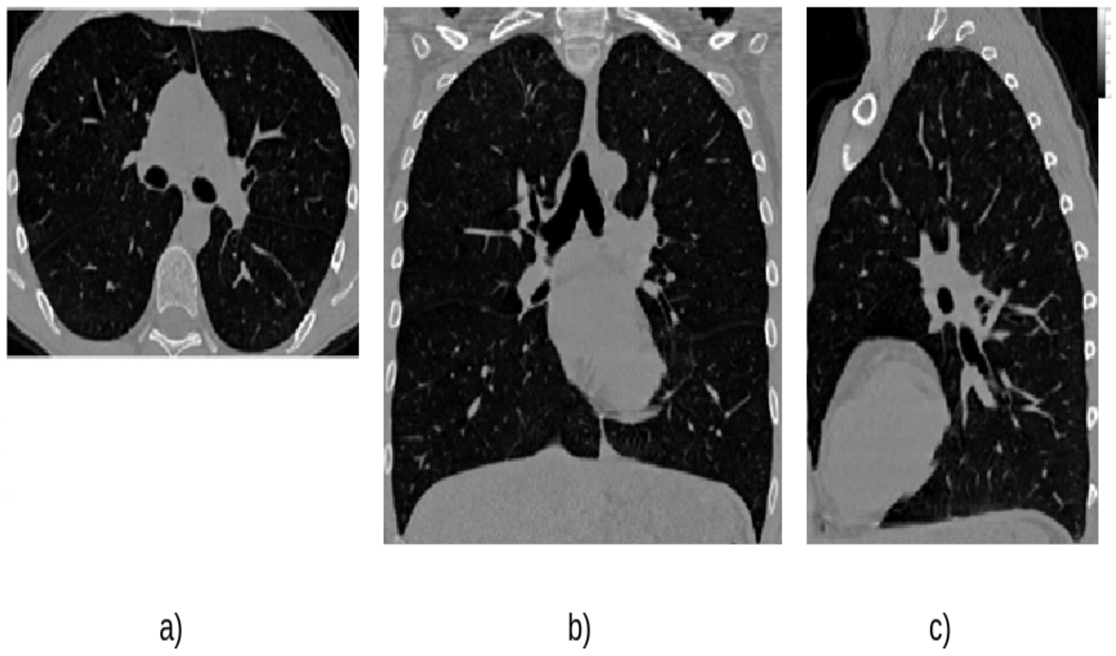

Figure 3.10: Figure showing different views of lungs CT scan [32] (a) axial, (b)coronal and (c) sagittal views.

of each voxel $0.97 \times 0.97 \times 2.5$ mm. In this experiment we have used 15 slices out of them, and resized it to 150×150 pixels i.e, it yields $150 \times 150 \times 15$. In order to generate moving images with different degrees of geometric distortions, we applied deformable or elastic distortion of order 0.8 [5]. We have generated 1000 distorted images for a single slice, giving a total of 15000 image pairs where the original slice (image) is considered as the reference and the corresponding distorted images are treated as its moving images. For the validation purpose, we split the dataset into training and test of 5000 and 3000 pairs, respectively. The Bayesian network is trained with 5000 image pairs using stochastic gradient descent and then validated using test data. We first show the visual results of using classical approach as shown in Fig. 3.11 then show the visual results of few challenging cases in Fig. 3.12 and Fig. 3.14

We now compare the reference image with estimated registered images using our approach as well as state-of-the-art approaches with different similarity metrics i.e, Normalized cross correlation [47], structural similarity index [1], Hausdorff distance [17], Root mean squared error [51] for the test data of 3000 image pairs. We first show visual results of classical approach in Fig. 3.11 followed by visual comparisons in Fig. 3.13. The calculated average values are as listed in the Table. 3.1 along with results using state-of-the-art registration frameworks such as selfsupervised FCN[30], DIRnet[50], and

Table 3.1: Quantitative results on lungs dataset.

Approach	NCC [47]	SSIM [1]	HD [17]	RMSE [51]
Ideal values	1.000	1.000	0.000	0.000
Classical approach [43]	0.632	0.210	900	45.80
Selfsupervised FCN [30]	0.824	0.412	980	45.2
DIRnet [50]	0.916	0.538	683	30.239
Voxelmorph [3]	0.921	0.532	632	28.438
Proposed (ϵ^{100})	0.939	0.623	652	**24.483**
Proposed (ϵ^{1000})	**0.947**	**0.632**	**580**	24.489

Voxelmorph[3]. Note that, ϵ^{100} and ϵ^{1000} are the means of 100 and 1000 samples from the posterior distribution . Since we are taking the mean of samples , the proposed approach with ϵ^{100} and ϵ^{1000} have shown better results when compared to existing state-of-the-art approaches. Fig. 3.15 shows the sampled weights from the learned posterior distribution when trained on lungs CT and cardiac MRI dataset respectively.

3.4.2 Quantitative results on cardiac dataset

In this subsection, we discuss results of classical, conventional deep learning and our approach on the cardiac MRI images[2] and compare it with state-of-the-art approaches. The dataset consists of short axis cardiac MR images and the ground truth of their left ventricles endocardial and epicardial segmentations. The cardiac magnetic resonance images are acquired from 33 subjects. Each subject's sequence consist of 20 time points and 8-15 slices along the long axis. We considered slices of 256×256 voxels of one subject and resized it to 150×150 pixels. In order to generate moving images with different degrees of geometric distortions, we applied deformable or elastic distortion of order 0.8 [5]. We have generated 1000 distorted images for a single slice giving a total of 10000 image pairs. For the validation purpose, we split the dataset into training and test sets of size 5000 and 3000 image pairs, respectively.

Table 3.2: Quantitative results on cardiac MRI dataset.

Approach	NCC [47]	SSIM [1]	HD [17]	RMSE [51]
Ideal values	1.000	1.000	0.000	0.000
Classical approach [43]	0.542	0.300	845	63.80
Selfsupervised FCN [30]	0.816	0.419	320	35
DIRnet [50]	0.941	0.436	439	16.23
Voxelmorph [3]	0.931	0.244	291	14.932
Proposed (ϵ^{100})	0.943	**0.415**	389	14.27
Proposed (ϵ^{1000})	**0.956**	0.467	**365**	**13.451**

We first show the visual results of few challenging cases as shown in Fig. 3.16 and We now compare the reference image with estimated registered images using our approach as well as recent state-of-the-art approaches with different similarity metrics i.e, Normalized cross correlation [47], structural similarity index [1], Hausdorff distance[17], root mean squared error for the test data of 3000 image pairs. We first show visual comparison results in Fig. 3.18. The calculated average values are as listed in the Table. 3.2 along with results using state-of-the-art registration frameworks such as selfsupervised FCN[30], DIRnet[50], and Voxelmorph[3]. Note that, ϵ^{100} and ϵ^{1000} are the mean of 100 and 1000 samples from the posterior distribution . Since we are taking the mean of samples , the proposed approach with ϵ^{100} and ϵ^{1000} have shown better results compared to existing state-of-the-art approaches. It is also found that the distribution of sampled weights after the training phase of both cardiac MRI and lungs CT scan is Gaussian distribution as shown in Fig. 3.15 .

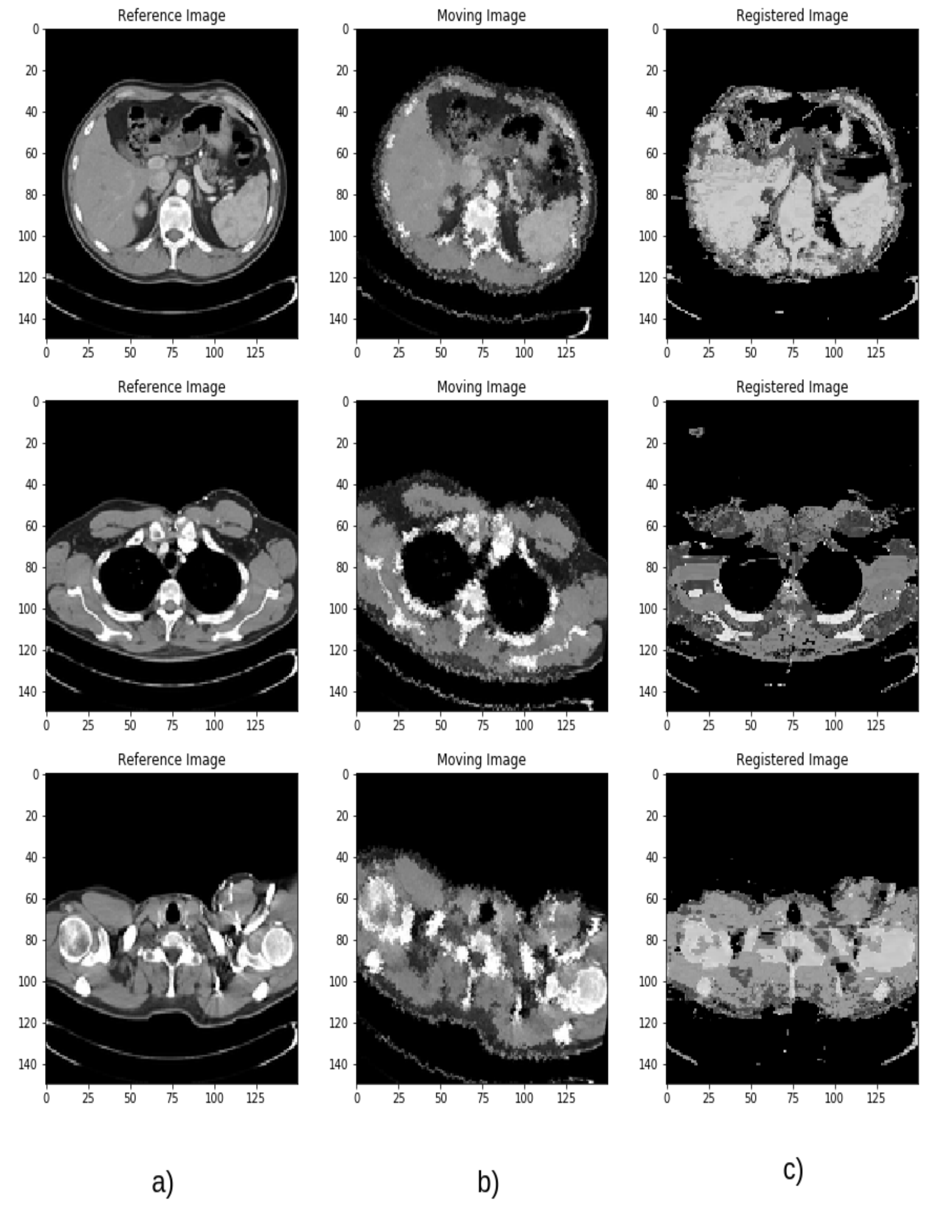

Figure 3.11: Visual results of lungs CT obtained by classical approach .(a) Reference image, (b) moving image and (c) registered image using classical approach [33].

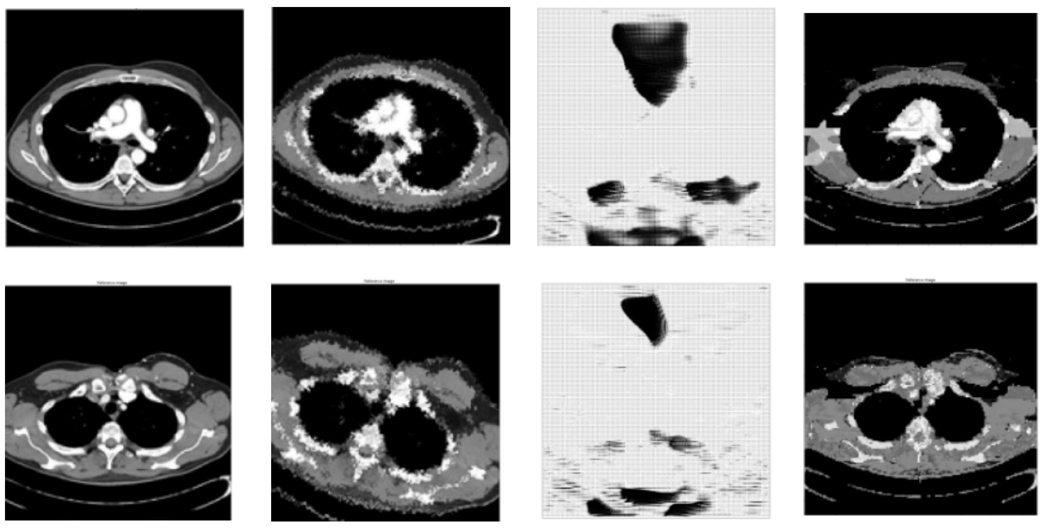

Figure 3.12: A few challenging cases in lungs CT dataset [8]. (a) reference images, (b) moving images, (c) estimated DVF, and (d) corresponding registered images using proposed Bayesian deep architecture.

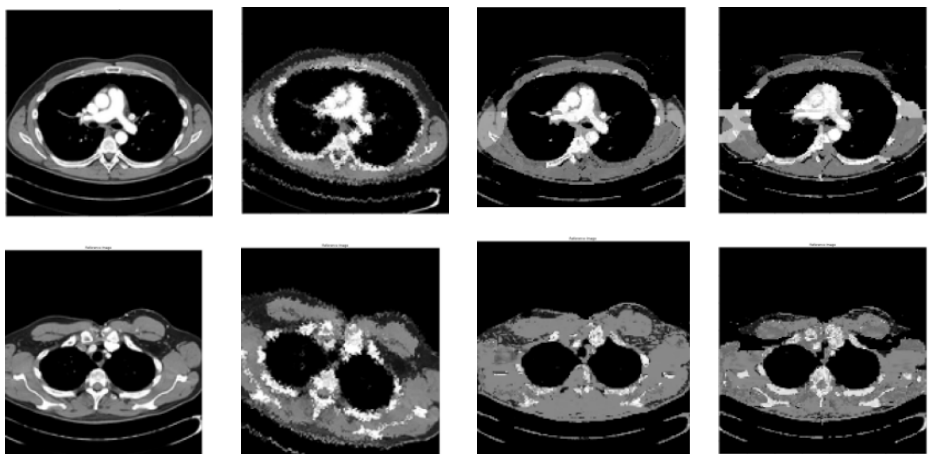

Figure 3.13: Comparative results on the challenging cases of lungs CT dataset [8]. (a) and (b) are reference and moving images respectively. (c) DIRnet [50] and (d) proposed.

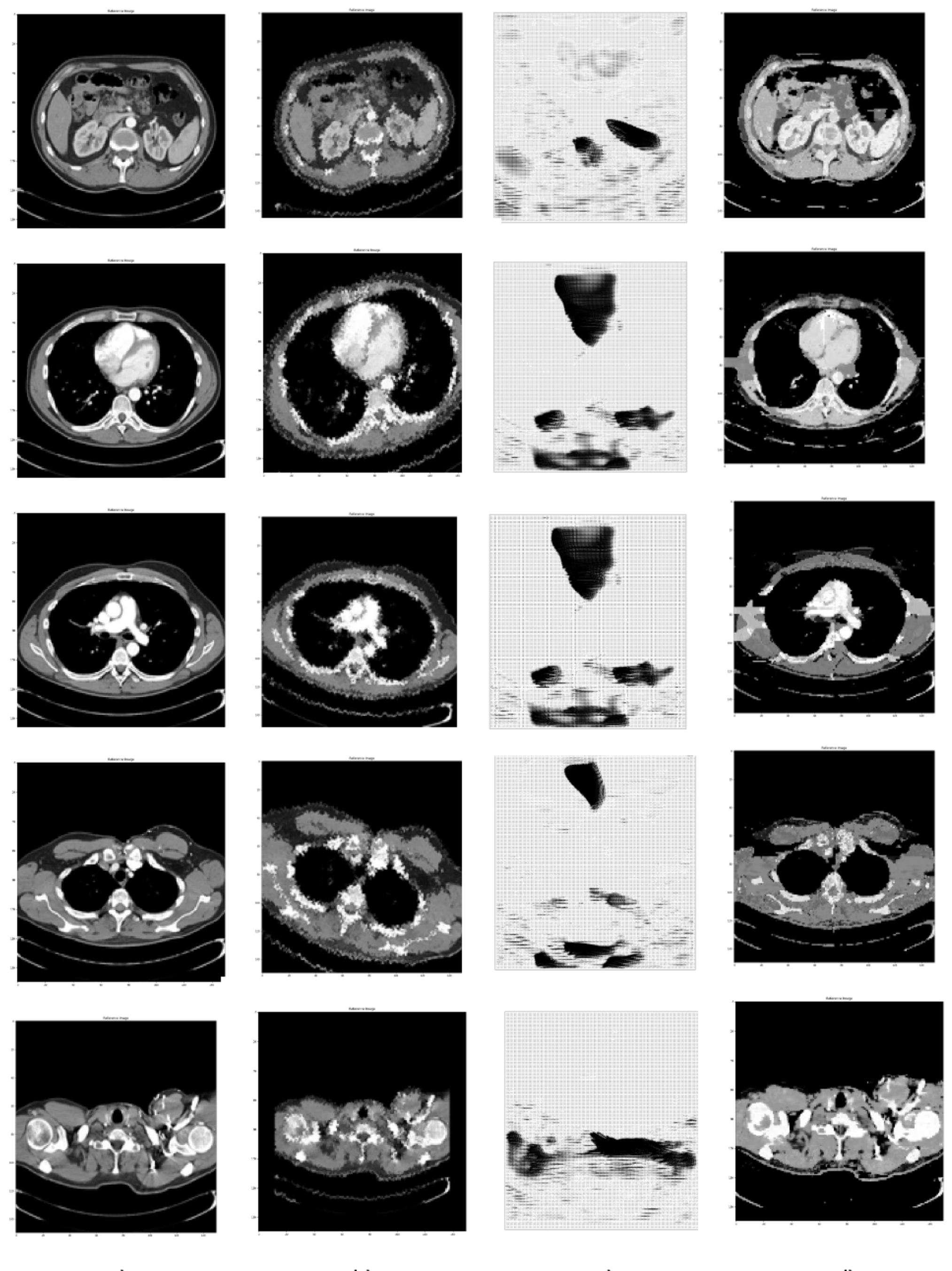

Figure 3.14: A few challenging cases in lungs CT dataset [9]. (a) referenceimages, (b) moving images, (c) estimated DVF, and (d) corresponding registered images using proposed Bayesian deep architecture

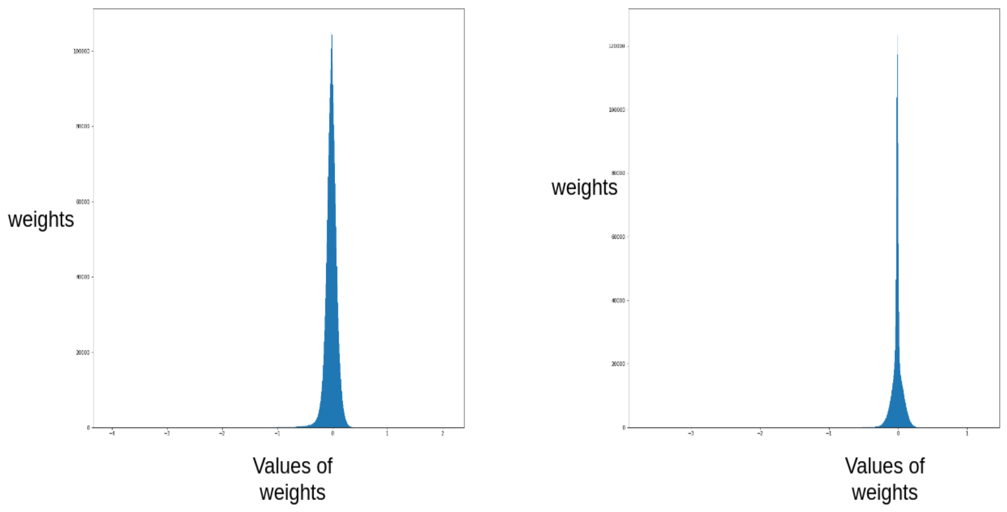

Figure 3.15: **Left:**Distribution of sampled weights from learned posterior distribution after training phase of Lungs CT. **Right:** Distribution of sampled weights from learned posterior distribution after training phase of cardiac MRI dataset.

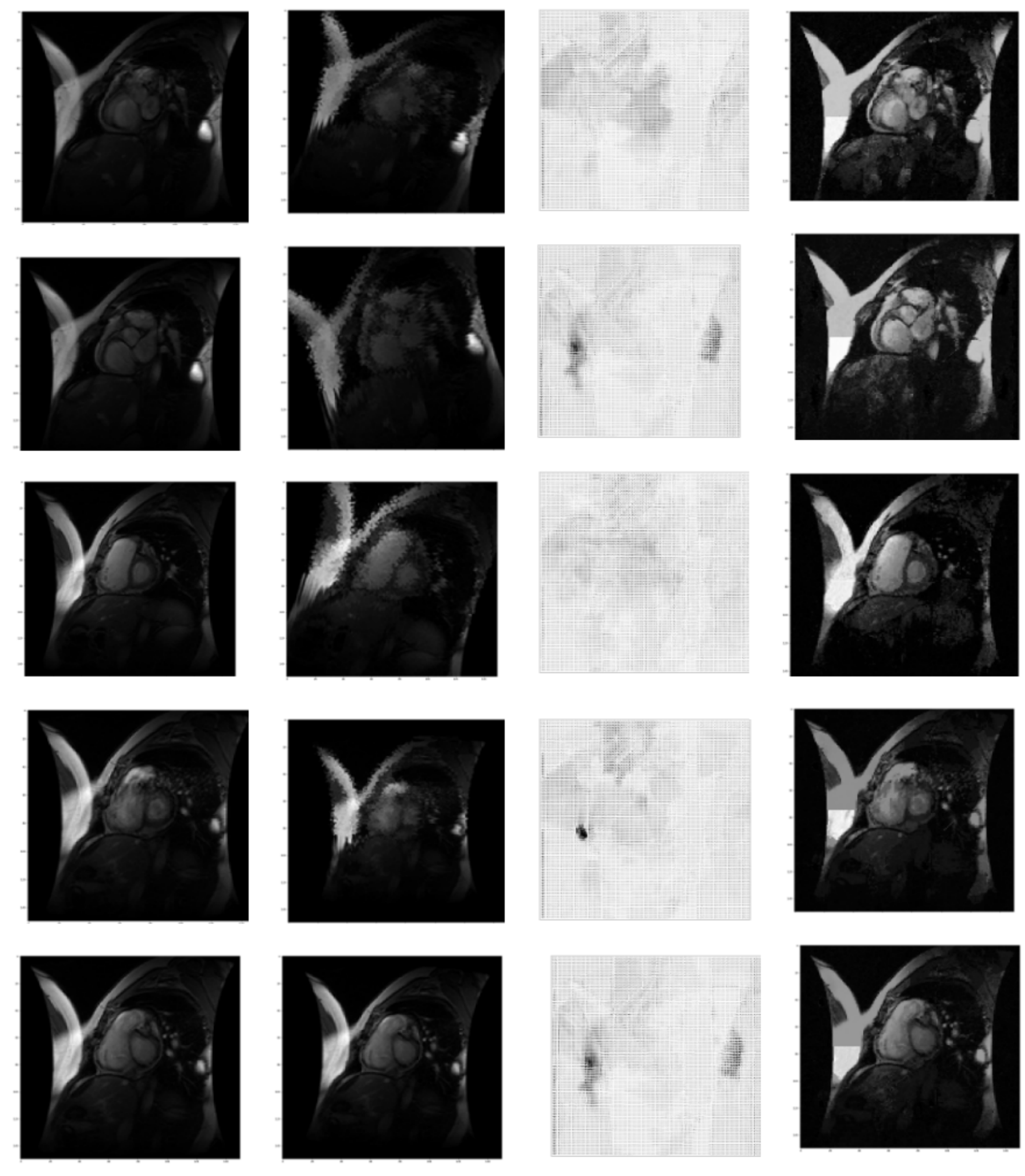

Figure 3.16: A few challenging cases in cardiac MRI dataset [8]. (a) referenceimages, (b) moving images, (c) estimated DVF, and (d) corresponding registered images using proposed Bayesian deep architecture

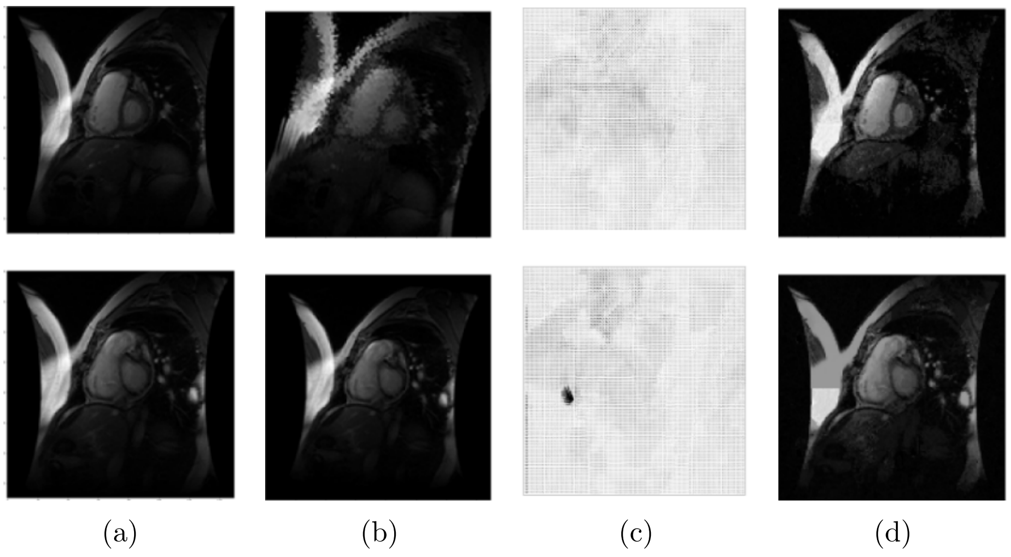
Figure 3.17: A few challenging cases in cardiac MRI dataset [2]. (a) reference images, (b) moving images, (c)estimated DVF, and (d) corresponding registered images using proposed deep Bayesian architecture.

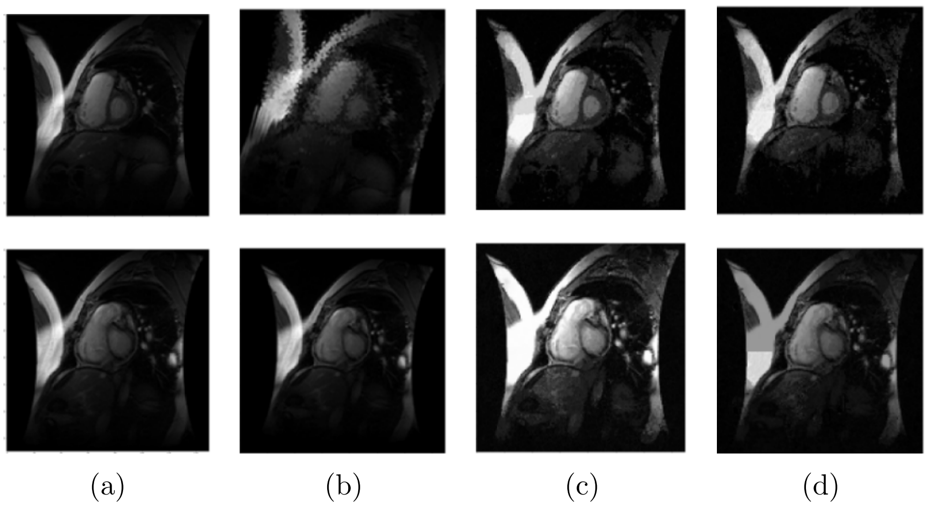
Figure 3.18: Comparative results on the challenging cases of cardiac MRI results [2]. (a) and (b) are reference and moving images respectively. (c) DIRnet [50] and (d) proposed.